TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

LASSI POHJOISVIRTA

CHOOSING A TOOL FOR IMPROVED SOFTWARE TEST MANAGEMENT

Master of Science Thesis

Examiner: Professor Miia Martinsuo
Examiner and topic approved by the Faculty Council of Faculty of Business and built environment on 14th August 2013

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Industrial Engineering and Management
POHJOISVIRTA, LASSI: Choosing a tool for improved software test management
Master of Science Thesis, 113 pages, 9 Appendix pages
October 2013
Major: Industrial Management
Examiner: Professor Miia Martinsuo
Keywords: Test management, tool support, challenges in test management, choosing software systems, build or buy decision

The purpose of this study was to investigate testing management activities of the case organization, find the most important features and criteria for the evaluation of different test management tool alternatives, and finally to assign merit for different alternatives by evaluating different tools so that the organization may select the most appropriate tool for them. The organization had two options: to acquire a commercial tool from an external vendor or to develop one themselves. Based on these objectives four research questions were assigned, for which giving answers would benefit both academics and practitioners.

The study is done in a manner that follows elements of action research and a single case study. It is a multi-method research combining several data collection methods: secondary data, semi-structured interviews, and observations and field notes. Test management is not a well-established practice in the academic literature, but it has become increasingly interesting subject. Software organizations aim to improve the efficiency and effectiveness of their testing activities, as it is very expensive and important part of the quality assurance of a software product. One way to do this is by implementing dedicated tool support around testing. There are various implementations of these test management tools, but they have in common that they facilitate performing test management activities with improved test asset management. The literature identifies certain practices when choosing software systems, but in test management tool context, this is rather a unique study in this regard.

It was discovered in the study that the case organization has various difficulties in test management activities mainly because of difficulties in test asset management. The most important features for the organization were the ability to plan, monitor, and log information within the tool, while the most important criteria included these functional and other non-functional criteria: usability, extensibility, and customizability. The evaluations showed that the commercial tools do fulfill functional criteria to some extent, but they have various limitations that team would need to cope with. Towards the end, an internally developed tool became increasingly beneficial option and it became the recommended alternative. The most important contributions based on the results of this study are both for practical and academic usages. The project and the organization where this study was done benefitted from the results as they gained information about their current processes and needs. The academic community gained a thoroughly documented case including difficulties in a software organization regarding test management, and an insight what kind of criteria are the most important when deciding which alternative is picked for test management solution.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tuotantotalouden koulutusohjelma
POHJOISVIRTA, LASSI: Ohjelmistotestauksen hallintaa parantavan työkalun valinta
Diplomityö, 113 sivua, 9 liitesivua
Lokakuu 2013
Pääaine: Teollisuustalous
Tarkastaja: professori Miia Martinsuo
Avainsanat: Testauksen hallinta, työkalu, haasteet testauksen hallinnassa, ohjelmistojen valinta, ”build or buy” -päätöksenteko

Tämän diplomityön tarkoituksena oli tutkia testauksen hallintaa kohdeyrityksessä, löytää tärkeimmät toiminnallisuudet ja kriteerit eri testauksen hallinnan työkalujen arvioimista varten sekä tutkia miten eri vaihtoehdot täyttävät nämä tärkeimmät kriteerit, jotta yritys voisi valita mahdollisimman sopivan työkalun heidän tarpeilleen. Kohdeyrityksellä on kaksi vaihtoehtoa: ostaa valmis työkalu ulkoiselta toimittajalta tai kehittää se itse omilla resursseillaan. Näiden tavoitteiden pohjalta työlle asetettiin neljä tutkimuskysymystä, joille vastaukset annettiin tavalla, jonka tarkoitus oli olla hyödyllinen kohdeyritykselle ja testausta koskevalle tutkimukselle.

Tutkimus on tehty yhdistelemällä elementtejä toimintatutkimuksesta ja tapaustutkimuksesta, ja sitä voi luonnehtia monimenetelmätutkimuksena, joka yhdistelee aineistokeruumenetelminä teemahaastatteluja, muuhun tarkoitukseen luotua dokumentaatiota sekä havainnointia ja kenttämuistiinpanoja. Akateemisessa kirjallisuudessa testauksen hallinta ei ole kovin vakiintunut toimintatapa, mutta siihen kohdistuva kirjallisuus on kasvanut. Koska testaus on erittäin kallis ja tärkeä osa ohjelmistotuotteen laadunvarmistusta, ohjelmistoyritykset yrittävät parantaa sen tehokkuutta ja oikeellisuutta kaikin tavoin. Yksi tapa tehdä tämä on ottaa käyttöön työkalu testauksen hallintaan. Testauksen hallintaan on kehitetty useita erilaisia työkaluja, mutta yleisesti ne helpottavat testauksen hallinnan toimintoja parannetun testausvarantojen hallinnan avulla. Kirjallisuus lisäksi huomioi tiettyjä käytäntöjä ohjelmistotuotteen valitsemiseen, mutta testauksen hallinnassa tällaista tutkimusta ei ole vielä tehty.

Työssä saatiin selville, että kohdeyrityksellä oli useita haasteita testauksen hallinnan toimintatavoissa, koska testausvarantojen hallinnassa oli ongelmia. Näiden haasteiden perusteella työkalutukea haluttiin parantaa. Tärkeimpinä toiminnallisuuksina työkaluille oli mahdollisuus suunnitella, valvoa ja kirjata tietoa työkalulla, ja tärkeimpinä kriteereinä oli näiden toiminnallisten vaatimusten lisäksi laatuvaatimuksina käytettävyys, laajennettavuus ja muutettavuus. Työkalujen arviointi näytti, että valmiit työkalut täyttivät toiminnalliset vaatimukset melko hyvin, mutta niillä oli rajoitteita, jotka vaativat muutoksia nykyisiin prosesseihin. Lopulta sisäisesti toteutettu vaihtoehto vaikutti parhaalta vaihtoehdolta ja sitä ehdotettiin käyttöön. Kohdeyritys hyötyi tuloksista, sillä se sai lisää tietoa sen prosesseista sekä tarpeista testauksessa ja sen hallinnassa. Työ lisäksi antoi tiedeyleisölle hyvin dokumentoidun tapaustutkimuksen tutkimusta varten, ja tulosten tarkastelut toivat ainutlaatuista näkökulmaa ohjelmistoyritysten haasteisiin testauksen hallinnassa sekä millaiset kriteerit ovat tärkeimpiä kun valitaan ohjelmistoratkaisua testauksen hallinnan.

PREFACE

This study has been quite a teaching experience. Writing this thesis has taught me much about doing research in practical settings. Doing the study, I noticed that there are various limitations that need to be coped with, but it can be truly fulfilling as the results have real meaning in the case organization. I have learned that in the end, it is the people in the industry that create the new best practices, and there is a lot of knowledge that can be used in academic research. I am sure I can take this experience and apply it to any future research needs.

Many people were part of the process getting this text into these black covers. I would like to thank my examiner prof. Miia Martinsuo for her help with setting the research questions, defining the scope, and other helpful tips during this project. Furthermore, I want to thank Minna Vallius from M-Files for her reviewing seem-to-never-ending iterations of my work and feedback, and Eija Vaittinen, for her relentless reading of my last versions and propositions for modifications. Finally, thanks to everyone at M-Files who have participated in making this study come true; it would not have happened without you.

Lassi Pohjoisvirta

In Tampere,
20 October 2013

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

API	Application Programming Interface
COTS product	Commercial-off-the-shelf product
CSV	Comma-Separated Value
IEEE	Institute of Electrical and Electronics Engineers
ISTQB	International Software Testing Qualifications Board
UML	Unified Model Language

1 INTRODUCTION

1.1 Background

Software testing is done for every software product that has any need to work properly. Testing is a complex and essential activity fully integrated to the software development processes. Partly because of its complexity, testing has many challenges (Bertolino 2007), while in the industry it does not get the resources it might need (Kasurinen et al. 2009). Companies may not be ready to make additional investments on testing efforts, and instead they strive to reduce the costs of software testing while at the same time improve the quality of their product (Majchrzak 2010b). To accomplish this, companies need to become more effective on their testing practices. One step towards more efficient practices, one could argue, is to automate difficult or laborious parts of the testing process. Automation in testing context with tools is not a new concept in the industry. Even though many companies have started to adopt test management tools (e.g. Eldh et al. 2010; Parveen et al. 2007; White et al. 1993) and there are various commercial tools available in the market, a tool to facilitate test management may not be as popular topic in software community as, for instance, automating test execution. It has been noted that many test management tasks can be automated with tool support and doing it has various advantages (Eldh et al. 2010). On the other hand, automating everything is not feasible or even desirable, so companies need to know exactly what needs to be automated.

As one of the ways to become more effective in processes is to automate, dedicated tool support is needed. Commercial tools for test management are numerous, and therefore choosing a software product from long list of options is no trivial task. Wrong decision on tool may result in significant financial losses to the organization (Jadhav and Sonar 2009), and evaluating software has been noted to be a difficult process (Carney and Wallnau 1998). Therefore it may be argued that it is imperative to make the decision based on well collected information. In order to do this, one needs to know what the tool is for, how to evaluate and compare possibly very similar products against each other.

One of the organizations aiming to fully utilize testing resources and gain better traceability in its testing activities and between testing assets is a Finnish software company M-Files Corporation, which in this study will be referred as “the case organization”. At the beginning of the year 2013, due to some difficulties and limitations of their current test management systems, M-Files started internal project aiming to acquire the most suitable test management solution for their organization. This project involved gathering information regarding test management and tool support for it. It was believed by initia-

tors that investment on tool support would both increase traceability and testing reporting capabilities, both of which had started to become increasingly important for the organization. This thesis will set the research questions based on this case project, while providing the necessary information for the tool choice decision.

1.2 Objectives and research questions

The main purpose of this study is to increase understanding on test management, current difficulties in it, and guide the acquisition of a test management tool in the operative setting of the case organization. Ultimately this study will shed light on how test management activities can be improved by test management tool and what considerations need to be made when choosing the most suitable alternative for the organization. To improve, in this study, is to make the activities more efficient and more traceable. These aspects will be discussed in a manner that academics can also benefit from the results. The most interesting question in practical perspective is which tool the organization should choose for its needs. Although the answer will not be explicitly included in this study, it provides valuable information for the people doing the final decision. Based on the project setting introduced in the previous section, the following research questions are defined for this study:

1. What kinds of difficulties does the case software company have in managing its testing process?
2. What features should a tool for test management include so that it will help with the identified difficulties and performing test management activities?
3. What kinds of criteria are the most important in choosing a test management tool?
4. How do different alternatives fulfill these specified criteria?

Answering these research questions is done by reviewing the literature and analyzing empirical data collected in the case project. The first question aims to seek information about the difficulties in managing testing in general and in the environment of the case organization in order to locate the problem areas for which the tool may be used. Answering the second question will shed light on what features a tool for test management should include and how automating parts of the test management process with a tool may help with the difficulties identified in the previous research question and performing other test management tasks. The third research question aims to provide information to the problem regarding what considerations need to be made when choosing a test management tool when the organization can choose between developing the tool internally and buying a readily made solution from external developer. Finally, the fourth research question aims to increase understanding on how commercial tools and internally developed tool fulfill the features and other criteria set for a test management tool.

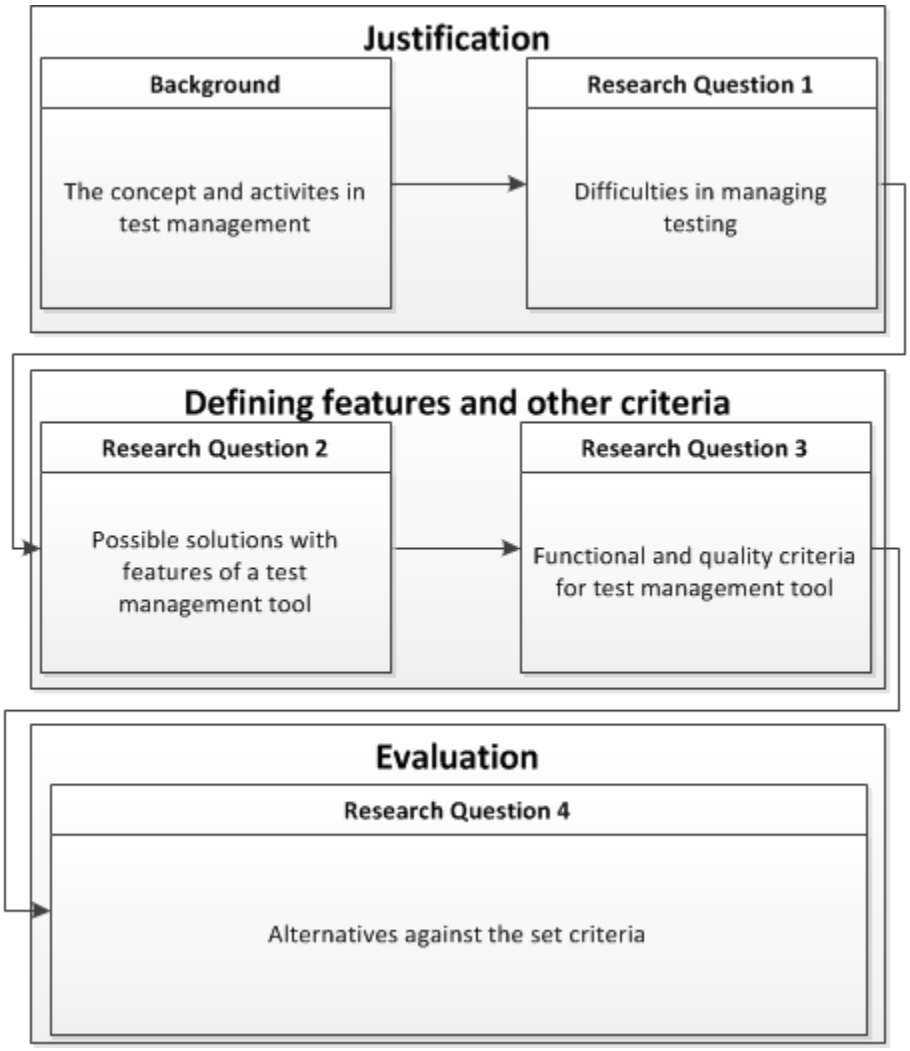


Figure 1: The connections between research questions.

Although the study aims to answer the set research questions separately, they are closely connected to each other. The structure of this thesis will follow the order of these research questions roughly in the Literature review part of the thesis as well as in Results and Discussion parts. The set research questions may be roughly categorized into three phases: justifying the need for test management tool, lining considerations and desired attributes for test management tool, and evaluating available alternatives. These phases and connections to the research questions and elements produced when answering the research questions are visualized in Figure 1. The progression of this study and the rationale behind the research questions may be described with these categories, as it shows the connections and order between the questions. In first of them, justification, test management and problems in it are introduced to provide information why organizations are seeking tool support and to answer why it is needed. Desired attributes group collects all the information regarding what aspects of test management tool support is aiming to facilitate, i.e. the functional and quality properties a tool should have. For this, one needs to know what the organization is trying to automate, i.e. activities in test management, and what solutions may be implemented to facilitate performing activities. The third and final group, evaluation, will discuss how the alternatives fulfill these de-

sired qualities and considerations for a tool. In order to do this, there needs to be a set of criteria to evaluate against. These groups will be used in Discussion chapter in which the research questions are answered.

1.3 Delimitations and structure

As ultimately this study aims to provide information regarding test management tools, the perspective of this study will be more technical and organizational. Other aspects of test management, such as leadership or change management, will be omitted from this study. In addition, although the themes of this thesis are closely related to test process improvement, or TPI, this thesis will focus more on test management tool and software system evaluation aspects than how improvements should be implemented in an organization. Test process improvement aspects are therefore considered secondary to this study. Similarly test execution techniques, i.e. how tests are executed, although very relevant to testing process, are not part of this study.

This work contributes in various ways to the current academic literature. Firstly, the academic literature does not describe the process of choosing a software system in practical settings very often. In very specific areas, such as selecting specifically a test management tool, the process of choosing a system has even less coverage in literature. The rare case studies relating to test management tools, such as ones from Parveen et al. (2007) or Safana and Ibrahim (2010), more often describe how the tool is implemented in organization after the tool has been chosen. This kind of literature, as well, is scarce even though there are numerous test management tools available in the market. This work will make contributions to these gaps in the literature, providing insight into decision process in SME context as well as discussion including what the most important considerations are in the case organization setting. The analysis made in this study may be used to further research what the needs of test management are, what activities should be supported by a tool, and how the evaluation of different tools can be done.

The structure of this thesis follows rather a typical academic piece. At the beginning of each chapter the structure of it is briefly introduced. After this introduction in chapter 2 the research methodology of this study is introduced. In chapter 3, the literature is reviewed to acquire understanding about the research questions for empirical research. The results of the information gathering project are reported in chapter 4. In chapter 5, the research questions set for this study are discussed based on the literature and empirical evidence. Finally, in chapter 6 the research is concluded with the major findings and discussion regarding limitations and contributions.

2 RESEARCH METHODOLOGY

This chapter introduces the background and choices made for this study in order to answer the set research questions. The limitations of these approaches are discussed briefly, as well. In the first section the research strategy and background is described to show how the study is done and why it is done in this manner. In the second section of the chapter, the different data collection techniques and analysis methods are introduced.

2.1 Research strategy

The research strategy used in this study has themes of what literature refers as a single-case study. A case study may be regarded as a strategy which involves empirical investigation of a phenomenon in real life context and uses different sources of evidence (Saunders et al. 2009, pp. 145-146). Yin (2003, p. 5) recognizes that one of the characteristics of a case study is that it does not require control of the behavioural events but rather requires the phenomenon to be contemporary. He also argues that the rationale for using single case study should be also clear. In this study the rationale for using single-case study is driven partly by practical reasons; there were no other known opportunities to follow the process of choosing a test management tool at the time of the study. Furthermore, using multiple case study might have limited the depth of the analysis, and the observations could not be made as thoroughly as done in this study. The study may additionally represent what Yin (2003, p. 41) may call a typical case; the case may be regarded a normal software system acquisition project among other projects, and the lessons learned in it may be used in similar projects. However, the context being a rather unique, as in specifically revolving around test management tools and the rather unique process of the case organization, the number of future projects that can use the lessons learned as they are may be limited. In research methodology literature, it has been noted that generalising based on a single-case study is not advisable, and the need for generalising the findings would require the use of multiple cases (Saunders et al. 2009, pp. 146-147). However, in this context it may not be a problem, as the goal of this study is increase understanding specifically in the case organization context and not generalise it extensively beyond that.

Additionally to being a single-case study, this study includes elements that are associated with action research. Saunders et al. (2009, pp. 147-148) recognize that typically in action research the researcher is closely integrated to the organization in which the change is happening, the research is iterative with context and clear purpose, and it should have implications beyond the project. As the researcher is employed by the case

organizaiton during the time of this study, the involvement in the organization comes naturally without extra effort. The project may be also regarded as iterative and evolving during the project, as the outputs of tools further specify the needs for the research, and they may not be specified completely before the project. The purpose or aim of the study, however, is clear: to investigate the difficulties in managing testing, see how test management tool may help with these problems, and finally set criteria and evaluate alternatives based on these criteria in order to get the best tool for the organization. The nature of action research and selected strategic approach gives the research questions room to evolve and find other interesting aspects for research. Furthermore, even though the research does not aim to generalise the results for use of other organizations or projects, it will try to provide suggestions and report the needs that are part of choosing a test management tool.

2.2 Data collection and analysis

The study will use multiple qualitative methods to collect data for answering the set research questions. Saunders et al. (2009, p. 153) point out that using multiple methods in the same project is advantageous when these different methods are used for different purposes of the study. In the context of this study, in which there are four separate research questions, the need for using multiple methods becomes apparent for answering them for various reasons. Answering different research questions, although they have also same data collection techniques, require different techniques because one method cannot cover all the questions. For instance, there are no available secondary data to answer what difficulties the company has, and they need to be gathered by interviews and observation. At the same time, different data collection techniques are used within one research question in order to understand the context from different perspectives. The interviews provide valuable data from different perspectives in the testing team, while observation notes may be used to further investigate the phenomena from the researcher's perspective. The latter may provide perspective that has the academic literature in mind, while the team has valuable experience how parts of the subject of this study are witnessed in the environment of the case organization.

The used data collection methods and their relations to different elements in this study are presented in Table 1. It can be seen from the table the pragmatist and multi-methodical nature of this study; all the methods used in the research help to answer multiple the research questions in some manner. Difficulties in managing testing, what features are required from test management tool, and what are the most important criteria for evaluation are not covered by secondary data, because primary data could be more accurate and there was no available data for this context. Observations, on the other hand, are partly used in every research question. The details of different data collection methods are presented in the following sections.

Table 1: Data collection techniques for acquiring information to different elements of this study.

	Background on test management activities	Difficulties in managing testing	Desired features of a test management tool	The most important criteria for the tool	Evaluation of the alternatives
Secondary data	**x**				**x**
Semi-structured interview	**x**	**x**	**x**	**x**	
Observation	**x**	**x**	**x**	**x**	**x**

The analysis of the data collected in this study can be characterized as a combination of deductive and inductive approaches, which are used for different purposes. Shortly described a study including deductive approach uses existing knowledge on the subject to "-- devise a framework to help you to organise and direct --" data analysis, while in inductive approach the data is first collected and then analysed to see what themes emerge from it (Saunders et al. 2009, pp. 489-490). In this study, the literature is used with every research question to specify terminology and major themes. The interviews are also influenced by the difficulties identified in the literature, although they are not included as questions directly. Furthermore, in the Results section of this study, the difficulties are grouped using the activities and elements specified in the existing literature in deductive manner. This way the study will show what kinds of difficulties are present in both the case organization and the literature so that they can be compared better.

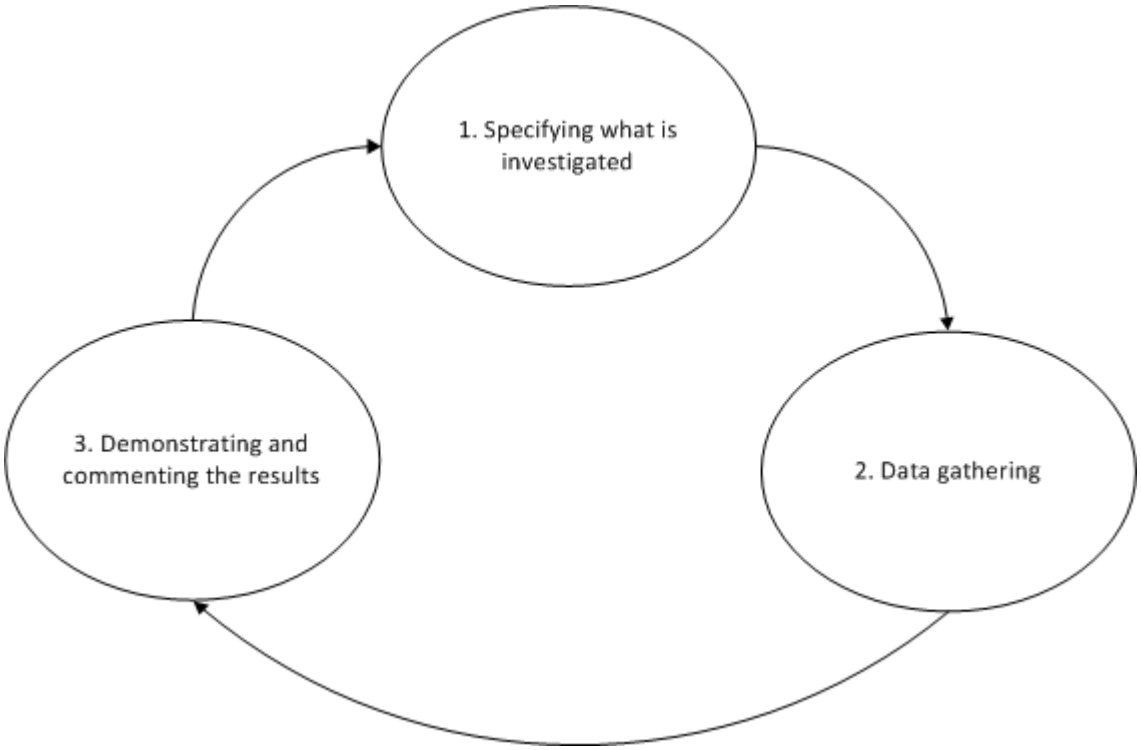


Figure 2: The iterative manner of developing desired functionality and evaluation.

On contrast to the deductive approach, analysis regarding desired features for a test management tool as well as criteria for evaluation does not take the academic literature into consideration. Furthermore, tool evaluation is done during the case project by combining different sources into a fact sheet, in which the data is grouped under feature groups that are desired in the test management tool and that are developed during the project. These parts of the study can be considered follow more inductive approach, as the categories and aspects that guide the research are developed during this study, not based on previous research. In the absence of literature regarding choosing test management tools, this was seen a way to provide insight in new context using existing terminology. Another reason for this approach was that the categorization of features or requirements was seen a natural way of describing a software system, as a new software system is developed from a set of features or requirements (e.g. Langer 2012, p. 6). These groupings are introduced later in Results chapter. The iterative manner of specifying the elements that are investigated, data gathering, and demonstrating and commenting on results is visualized in Figure 2. Specifying what is investigated and commenting the results are done when the whole team is present, while data gathering is done solely by the author.

2.2.1 Secondary data

This study will utilize secondary data from multiple sources. The two main reasons for using secondary data in this study are to get information about test management practices in the case organization and acquiring data for evaluation of different tools. The secondary data are listed in Appendix C with observation field notes.

Exploring the test management activities in the case organization and forming a context require investigating the documented procedures stored internally for the organization. These are used with the observations made by the author to describe the testing and test management practices in the organization. However, as the documentation is used for different purposes than to describe processes in the level desired in this study, they have to be used with some reservation. The documentation may describe procedures not completely used, or they may be out of date. Ultimately the observations, semi-structured interviews and secondary data in combination will provide the best understanding about the current activities in the case organization.

The fourth research question of this study involves evaluating the selected tools for test management against the specified criteria. This is partly done by analyzing data gathered from secondary sources. Main secondary sources are the webpages provided by the vendors of the tools. These findings are complemented with the field notes made when evaluating the functionality with the people involved in the project. Analysis of data collected from secondary sources regarding test management tool functionality is done by collecting the data into a fact sheet. In these fact sheets the data is grouped by different features or aspects of the test management tool that are specified further during the project. During the reviews with the team, these results are discussed and further investigation needs are set until there is enough information to make a decision.

2.2.2 Semi-structured interviews

For this research, semi-structured interviews were chosen to find out the views of different people in the testing team and among immediate stakeholders. This interview type was rather an obvious choice, as they allow the interviewees to use their own words and ideas as well as lead the conversation to an unexpected but perhaps interesting direction (Saunders et al. 2009, p. 324). The interviewee, for example, may bring up some information related to this study that the interviewer may not have anticipated. Semi-structured interviews are also most suitable for situations when the questions are open-ended and the questioning may be varied in some way (Saunders et al. 2009, p. 324). In this study, it is not feasible to ask all the questions from all the interviewees, as everyone is not involved in testing in the same manner. Furthermore, in this study the purpose of interviews may be regarded to provide a formal setting where the central topics of the study can be discussed. The structure is set for the interviews to guide and limit the conversation to these topics.

The questions of semi-structured interviews are chosen to acquire information about the following themes:

- The tasks done for testing
- The difficulties in management of testing from the team's perspective

- The most important features and criteria for a test management tool among different stakeholders

The themes are selected to increase understanding of testing process and practices as a whole, its challenges and the most important features the team members think that the tool should have, and what criteria should be used to evaluate the tool. Even though the interviewer took notes during the interviews, these were also recorded and transcribed afterwards in order to validate the gotten information. The full structure of interviews is included in Appendix A.

As the testing team and the other stakeholders using the tool constitute a small group of people, there was little need for sampling, and every central employee was interviewed. In this study, the interviewees were two software testers that had experience in testing from multiple years, the test team manager, an employee formerly doing the tasks of test manager and software tester, and program manager, who supervises the quality of the product and processes. The list of individual interviews is included in Appendix B, in which notations used in results chapter are specified. These interviews were considered to be sufficient for this study, as these employees were direct users of the tool or the main benefactors of the tool. The other possible interviewees were new employees in the company, and they had no relevant experience in either test management or the testing processes of the organization.

Data gathered from interviews is analyzed by finding similarities and differences in difficulties in test management practices in the reviewed literature. The tasks done for testing are used to analyze the background of management of testing effort: how the different stakeholders are involved in the testing process. The difficulties noted by the interviewees will be used among the observation field notes with the reviewed literature to check what kinds of difficulties are recognized. Desired features and the most important criteria are categorized almost solely based on the themes emerging from the discussions and observations, not on any literature source. This "bottom-top" approach is relevant in this study as the evaluations are done based on these themes, and just analyzing the views of the literature by deduction would not be helpful to achieve the goals of the study.

2.2.3 Observation and field notes

In addition to the interviews and secondary data, the study is using observation as data collection method for all research questions. As the author has worked as a part of the testing team in regular basis during the time of this study, it is a great opportunity to observe the procedures of the organization as well as other situations, which will help to understand the difficulties in managing testing and functional requirements for tool support. Additionally, the author is handling most of the activities of test manager for a month’s time, which gives first-hand perspective on the tasks required for that job de-

scription. This is what Saunders et al. (2009, pp. 289-290) regard as participant observation, in which the researcher participates fully in the activities of subjects. Because of the need to understand the case organization context, observation and gathered field notes have a great emphasis in this study. During the empirical data collection phase of this study, which includes all testing phases in the organization, the author is observing the testing activities from the perspective of a tester and a test manager. All the discussions, exchanged e-mails, and meeting notes documented during this time are utilized, as well. These observations will be recorded either first on paper or directly to a computer file as simple, unspecified field notes, in which an observation is characterized by few sentences.

Firstly, the observations related to this study include test management activities in case organization as well as difficulties found by the observer or anyone else in the observed period. In the research literature, it has been noted that before one can take field notes, it has to be decided what should be annotated (Wolfinger 2002). In order to know what situations really relate to this test management, proficient knowledge of the topics of this study is needed. Therefore the observation period takes place mostly after the literature review of test management activities has been nearly finished. Ideally this helps the observer to note salient situations. On the other hand this approach produces risk to see only the aspects noted in the literature. As Wolfinger (2002) notes, tacit knowledge and field notes have an exact link between them and observations are affected by the background knowledge of the observer. In field note strategy that includes taking salient information, he also notes that it is highly subjective due to the tacit knowledge regarding what is salient and what is not. However, these may be regarded acceptable limitations, as without necessary knowledge of the study area, the observations may lack the information that is really needed, and comprehensive note-taking may be highly demanding for the purpose of this study. As the observations may be short and not come every day, and the timestamps may not be relevant for this study, they are recorded only in one file.

Secondly, the data gathered from trial versions of various alternatives are used as field notes. These are what Saunders et al. (2009, p. 296) may call primary observations or experiential data, in which the data is gathered from what the observer has witnessed happening. The functionality of the tools is such observable data: how it seems to work in different situations. This may be regarded a common way of gathering data when collecting information about a tool, although it may not be described part of observation data collection technique. This data may be used for investigating how the tools would help with difficulties and enhance other test management activities. Additionally the data is used as basis for evaluating the tools against the set criteria.

Observation field notes of test management activities and difficulties executing them are analyzed both deductively and inductively by finding connections and contradictions

between the gathered data and the reviewed literature. The results are validated by other employees whenever possible to maximize the validity. Observation field notes regarding evaluation are used with secondary data to form the fact sheets. The analysis on these observations is done by describing how the tool fulfills the feature categories that are developed during the project. Ideally the observations provide enough data in order to evaluate and compare different alternatives. However, there may be threats to validity and reliability when using this method. During the limited time of this project, some features may be overlooked or insufficiently covered if the functionality is misinterpreted. For example, the observer may see the tool lacking a certain mandatory functionality, even though the problem may lay on him for not knowing how to use the product. Even though in these cases the observer contacts the vendor, there may be other situations that he takes for granted and does not explicitly ask for them. This may be regarded an acceptable limitation, though, as the use cases set for tools may provide enough valid data for analysis.

3 LITERATURE REVIEW

The literature review of this study is divided into five parts. The first part, which includes an overview to software engineering and testing especially in a setting with evolutionary products, will shed light on what kind of environment the target company operates in and what kind of issues it has in terms of creating quality products. In this part, some of the terms and processes of software engineering used later in this thesis will be briefly explained, as well. The latter parts will provide a basis for this research as well as will try to see how the current research answers the research questions laid for this thesis.

3.1 Software testing as part of software engineering

The software industry is known for its many present and past problems of keeping its budget and quality of products and projects in desired or even in acceptable level. In the literature, this phenomenon has been called software crisis, which has started from the 1960s (Majchrzak 2010a) or 1970s (Agarwal et al. 2010, p. 13). The noted problems in software crisis include schedules and cost estimates being very inaccurate, the productivity of software people being not in pace with the demand, quality being insufficient, and maintenance tasks taking the majority of all software funds. Causes of the software crisis include a poor level of communication and that after delays in any process or stage, for example in testing, the scheduling does not match the realized timing. (Agarwal et al. 2010, p. 14) Majchrzak (2012, p. 4) states that problems in numerous failed projects could have been detected by testing the released software before usage.

Software engineering is a discipline that aims to introduce engineering disciplines into software development. A generic definition by Institute of Electrical and Electronics Engineers (IEEE) would regard software engineering as "—the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software, i.e., the application of engineering to software." It is driven by set of principles, methods and techniques, methodologies, and tools. Software engineering processes can be divided into four commonly used fundamental process activities: software specifications, software development, software validation, and software evolution. (Agarwal et al. 2010, pp. 9-10; 19) More broad boundaries are given by ISO standard 19759:2005 (Abran et al. 2004), of which disciplines in software engineering are gathered in Figure 3. This study will deal mostly with software validation or software testing, in which the software is validated to ensure that the software product works as intended (Agarwal et al. 2010, p. 19) and software engineering tools and methods, which

includes the investigation of various tools that allow repetitive, well-defined actions to be automated (Abran et al. 2004). It, however, can be seen to have connections to other disciplines, such as software engineering management and process from testing perspective and, of course, software quality, but as the boundaries are fuzzy, it would be simpler to limit the used terminology to software testing and software testing tools.

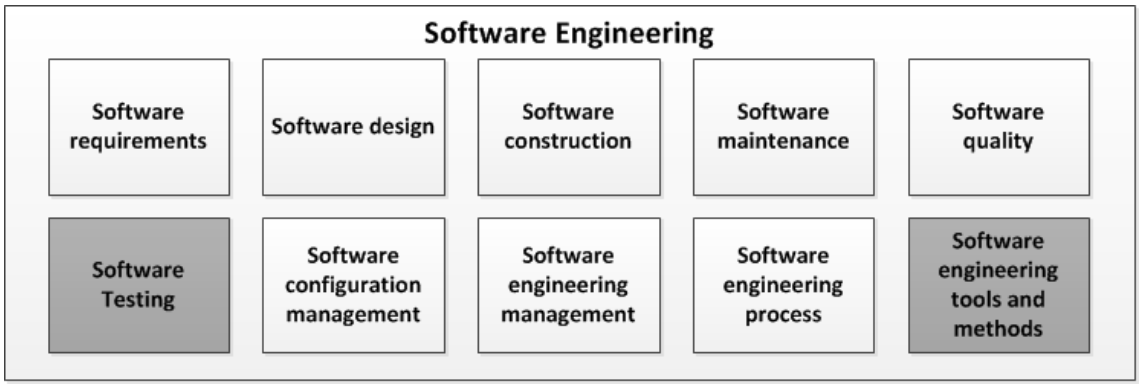


Figure 3: Software testing and software engineering tools among the other software engineering disciplines. Adopted from (Abran et al. 2004).

Software testing may be considered as a set of planned and systematically conducted activities (Agarwal et al. 2010, p. 161) that are "-- performed for evaluating product quality" and to improve it by identifying problems and defects against expected behavior (Abran et al. 2004). In more practical sense, software testing can be seen as a way to answer various fundamental questions regarding the developed software product, such as "is it working?" or "can we use it?" (Farrell-Vinay 2008, p. 9). Testing has been viewed as essential and primary activity to assure quality of software products (Bertolino 2007), though not without controversy. According to Black (2002, p. 408), testing can be seen as regrettable necessity and as the best of the bad alternatives to ensure the appropriate quality of software product.

Testing has been fully integrated to software engineering (Parveen et al. 2007) and is also most effective when it is conducted in parallel with software development (IEEE Computer Society 2008). Therefore testing is an activity that must not be ignored or left only to the end of software development. However, testing seems to be often overlooked and under budgeted. Understanding the nature and commonly used practices of software testing is fundamental in this study, as the evaluated tool is set to work in that environment. Hence, the concept and some widely used practices in software testing are introduced in the next few paragraphs.

In addition to validating does a new product fulfill its requirements, testing has an evolutionary side, as well. Brooks (1987) has noted that successful software gets changed for two reasons: for the product is found to be so useful so that the people want to use it beyond the original domain and for the product has survived the initial environment it was made for and needs to be updated to match the new environment. While great soft-

ware will evolve beyond its initial purpose and environment, it may be challenging in the terms of testing. The changes made to the original software may change the software greatly; something may be added to the product and something may be taken out of it according to business needs (Yang and Ward 2003, p. 3). Being a complex system, the relationships between different elements inside are not clear. Regression testing is testing that is done after there are changes done to the existing software system. The purpose of regression testing is to make sure that the changes made to the system have not created any unwanted functionalities, also called regression errors (White et al. 1993), to those parts that have not been changed (Yoo and Harman 2007). In this technique no more tests are created, but instead the existing sets of tests in form of test cases are chosen, prioritized and executed (Naik and Tripathy 2008, p. 17). It is important to decide how many test cases are chosen for testing and which of them are the most important, as regression testing is expensive and takes most of the testing effort (Naik and Tripathy 2008, p. 17). It is noted, though, that some companies run all the test cases in their regression testing without prioritizing them at all (Onoma et al. 1998). Finally, as regression testing mainly uses existing test cases, it may have a greater role in situations where the company is invested in certain developed products. Indeed, regression testing is seen especially beneficial to a company that develops similar products (Onoma et al. 1998).

The software testing discipline may be further divided into different focus categories. ISO standard separates testing techniques from other aspects of testing, such as testing process, testing measurement, and testing levels. These techniques include all the systematic ways of revealing defects, such as ad hoc testing or exploratory testing, while testing levels regard different levels of detail in testing, testing measurement revolve around gaining data from the system under test, and test process involve integrating concepts, strategies, techniques, and measures to an entity that is run by a testing team. (Abran et al. 2004). In his book, Majchrzak (2012, pp. 11-49) similarly divides software testing into testing techniques, organization of testing, and tool support. The introduced two set of categories interlace: organization of testing includes testing levels, testing process, and testing measurements, while testing tool support is regarded as part of engineering tool support. However the grouping is done, common themes exist in both of them, and using the terms used by Majchrzak (2012, pp. 11-49) they can be characterized as organizational and technical. As this study does not involve testing techniques, only organizational aspects are further introduced.

Different ways to describe organizational aspects of testing is through the terms test system (Black 2008, pp. 73-74) or test organization (Majchrzak 2012, p. 39). Black (2008, pp. 73-74) defines this concept as the organization's capability to test software, which is enabled through testing process, testware, and test environment. Testing team will operate in this system interacting with different elements of the system, using, maintaining and improving them. According to Majchrzak (2012, p. 39), organizational

dimension of testing is part of successful testing, comprising of "seeing testing as a process", dividing it into phases, documentation, test management, and testers' certifications. Again, the groups interlace, as testing process includes the phases in testing and documentation. For clarity reasons, the organizational aspects introduced in this study are test process, test environment, testware, and test management. As stated in delimitations section 1.3, this study concentrates more on organizational aspects and tool support than testing techniques. The elements described in the literature are visualized in Figure 4. Background on these elements is provided in the following paragraphs. Test management does not have as clearly set boundaries as other elements, and as a central part of this study, it will be covered more thoroughly in section 3.2.

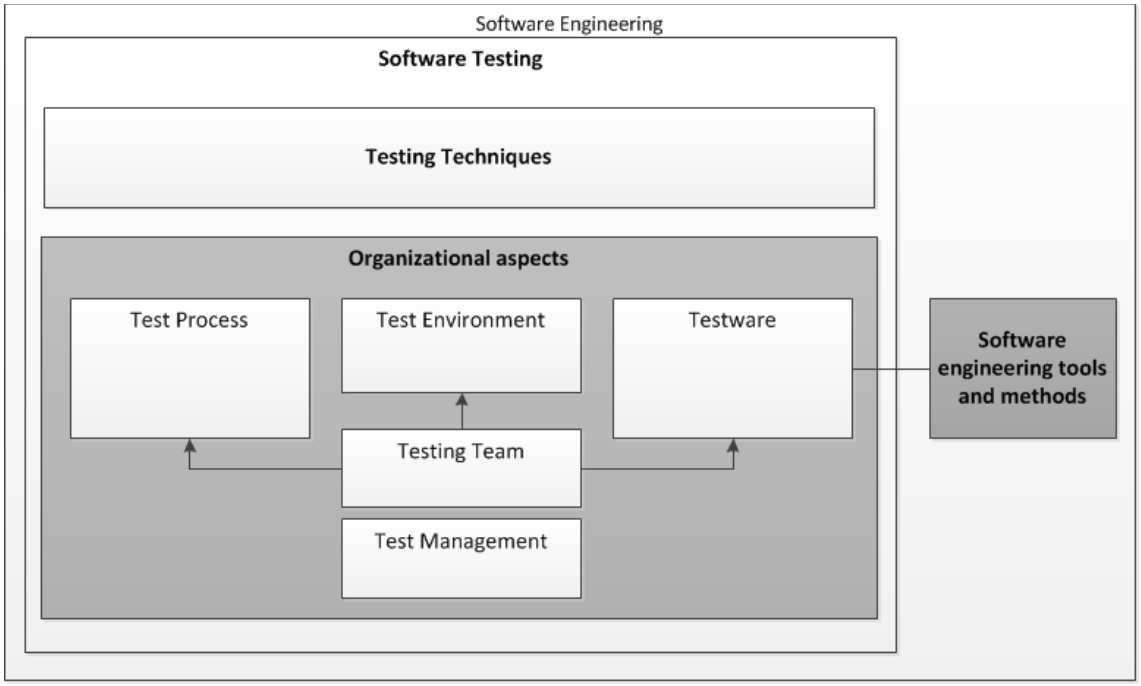


Figure 4: Organizational aspects of software testing separated from testing techniques.

In more general context, ISO 9000 standard defines a process as a "set of interrelated or interacting activities, which transforms inputs into outputs --" and needs resources, monitoring and measuring. Effectiveness of the process is how it is able to produce desired results, while efficiency of the process is the amount of results compared to the resources allocated to produce them. (ISO 2008) This may be applied into software testing process, as well; effectiveness of testing process is to produce reliable information about quality of system under test, and efficiency is to produce this information with reasonable resources.

In testing literature, testing process is stated to include all the documented and undocumented procedures to execute testing tasks in an organization (Black 2008, pp. 73). Farrell-Vinay (2008, p. 15) notes that the processes and standards within organization become more important when the number of employees becomes greater. Testing process may consider different phases or tasks. According to Majchrzak (2012, p. 39), in order

to have effective and efficient testing, the process must adapt to the status of developed program, and it needs to be done in phases. Testing is not done only once for the developed product, and usually the testing related literature includes different levels as phases for testing process. Often the literature recognizes several levels, such as unit testing, integration testing, and system testing during the development of a software product (Majchrzak 2012, pp. 39-40; Agarwal et al. 2010, p. 165; Naik and Tripathy 2008, p. 16). Additionally some might consider acceptance testing done by the customers one testing level (Naik and Tripathy 2008, p. 16), which on the other hand sometimes is regarded as part of system testing (Agarwal et al. 2010, p. 172). Other mentioned levels are functional testing, reliability testing, usage testing, stress testing, and field testing (Onoma et al. 1998). Testing is focused on different parts of the system depending on the level. In unit testing single components, or units, of the system are tested separately in order to verify that they work correctly (Agarwal et al. 2010, p. 165; Naik and Tripathy 2008, p. 16). In integration testing phase the subsystems formed by the components are tested in order to assure that there is no unwanted functionality when the components work together (Naik and Tripathy 2008, p. 16). Finally in the systems testing phase the whole system, as a combination of these previously tested subsystems, is tested to verify that the developed system fulfills the functional and non-functional requirements set for it (Agarwal et al. 2010, p. 172). After the developed product is tested and released, starts the last testing phase, maintenance. Majchrzak (2012, p. 41) note that maintenance phase is important especially in situations where the organization has periodic process and releases new versions of the program. The maintenance process includes the aforementioned regression testing.

On the other hand testing may be regarded as a set of sequential phases inside each level. Whittaker (2002) divides testing into four phases: modeling the software's environment, selecting test scenarios, running and evaluating test scenarios and measuring testing progress. Similar phases are listed by ISO standard, which includes activities planning, test case generation, test environment development, test execution, test results evaluation, problem reporting or test log, and defect tracking part of testing process (Abran et al. 2004). Both the sources note that in each level, the tests need to be planned and prepared, run, and the results need to be evaluated. These phases follow rather a similar pattern as the test management activities, which are introduced later in section 3.2.1.

Concepts of testware and test environment include all the concrete elements in testing. Testware includes all the used tools, documents and means for monitoring in testing (Black 2008, pp. 78). Test environment, sometimes called also test lab or test laboratory (Naik and Piyu 2008, p. 358), may be regarded as a collection of testing related tools (Farrell-Vinay 2008, p. 96) or everything needed to do the actual testing with the target system, for example computers, software, working space (Black 2008, pp. 73-74). According to Farrell-Vinay (2008, p. 60), test environment should include tools for man-

agement of defects, test execution, and test specifications. A good test environment can be seen to make testing improvement, test execution, and test management less complicated (Farrell-Vinay 2008, p. 96).

Testing team includes various employees with different roles. Based on various literary sources, Illes et al. (2005) have recognized different tasks of different roles included in testing process. Test manager identifies the test strategy, prioritizes test activities, assesses risks, does staffing decisions, defines metrics to monitor the progress of testing, and defines when the tests pass and fail. Test manager will also track the problems and defects in the tested software with software tester and developer. Test designer designs and constructs test cases to be tested. Tester will execute these test cases, collects the results and compares them and reports the results of these results. Finally, test configuration manager and test administrator will manage the test ware.

The main purpose of the whole system is to offer testing services effectively while recording central metrics of testing activities. Additionally the system should help testers focus on the relevant tasks, manage defects, and analyze the metrics gathered (Black 2008, pp. 73-74). Test system will require information as its input and provide information as output. According to Farrell-Vinay (2008, p. 60, 63) the input of a test system is an unambiguous description how the system under test should work, while the several output documents are test plan, test case specification, test reports, and release notes. IEEE standard 892-2008 defines these same documents for an output of planning and execution, as well. In the standard, it is noted that the documentation should be done for different levels, as in master level guiding all activities and specific documents for specific testing levels. (IEEE Computer Society 2008)

3.2 Test management

Management of testing has become part of the testing literature, and modern testing-related textbooks often include it as a topic (Majchrzak 2012, p. 44). However, there does not seem to be great body of research on the topic test management. As a term, test management has been noted to be a broad one, which cannot be defined easily and may include various tasks (Gao 2011; Liu and Robson 1992). The lack of academic research on the topic drives the study to use publications and books that may be based on personal experiences and adopted practices, which may not have sufficient empirical evidence to be very credible sources. When reviewing literature extensively like in this study, it is important to define the concept test management as the literature does not always explicitly recognize some of the tasks part of it. Without a clear definition, some of the observations may be overlooked due to lack of the knowledge is something really part of the subject of this study. Furthermore, the definition of what test management is and what it consists will be used as a definition what is a good test management tool. In the first section of this chapter, different aspects of test management are discussed from

literature point of view. In the following section, activities of test management are further investigated in order to understand what the activities are and what is needed to perform them.

3.2.1 Concept of test management

The literature, in which the term test management is explicitly used, defines it slightly differently from each other (Parveen et al. 2007; Davis 2006) or the term is used without any explicit definition (e.g. Majchrzak 2012; Louridas 2011; Farrell-Vinay 2008; Black 2008; Ramler 2004). In fact, the latter is very often the case even in textbooks specifically explaining managing testing effort, and even though the authors may use the exact term, an explicit definition for test management is rarely given. The reason for this may be that the term may be regarded as a simple one that does not need a specific explanation. Parveen et al. (2007) regards test management as "-- a method of organizing test assets and artifacts such as test requirements, test cases, and test results to enable accessibility and reuse." This emphasizes the organizational aspect of test management. The definition from Davis (2006) describes test management as "-- the practice of organizing and controlling the process and artifacts required for the testing effort --", which would include not only the artifacts as part of test management but controlling the testing process as well. Not having a commonly known definition may difficult grasping the concept what test management is.

If the definition for test management is not given in an explicit manner, one way to approach it is to see what tasks or activities are regarded to be included in test management process. This approach seems to be more popular in the literature, as well, although many authors settle for listing activities without more thorough explanation. IEEE standard 829-2008 describes test management process as a set of tasks regarding preparation of the plans for execution, monitoring the execution, analysis of anomalies, reporting progress of the test processes, assessing test results, determining whether a testing task is complete, and checking test results for completeness. It also includes identifying the process improvement opportunities in test execution. (IEEE Computer Society 2008) In addition to managing, monitoring and reporting involved in a test project, International Software Testing Qualifications Board (ISTQB) includes leading the testing team and testers and handling external and internal relationships as tasks in test management (ISTQB 2011). Black (2008) also introduces what kind of tasks are included in test management: creating documentation regarding managing testing, estimating time and work efforts in test activities, planning a schedule for testing, monitoring the progress of testing and communicating the business value of testing to other parts of the organization. Davis (2006) would also include coordinating the efforts of everyone involved in testing, tracking dependencies and relationships among test assets and defining the goals of quality as well as measuring and tracking them as tasks of test management. Finally, Louridas (2011) sees test management as a set of tasks dealing how test cases are found, organized and assigned to testers, and how the testing is organized, the

results collected and the progress monitored, emphasizing on the test asset organizational aspects of test management.

It can be seen that even though different sources list slightly different tasks as part of test management, the activities are rather similar; all the tasks revolve how the tests are prepared, monitored, and controlled as well as the results analyzed and reported. Few authors also highlight the test improvement to be part of test management. Additionally some sources would also include leadership aspects, i.e. the leading the test team and promoting the importance of testing to other organization units, in test management. These tasks seem to be organizational and managerial in nature, and they do not seem to include any test execution tasks.

Some authors would divide the test management to different phases or steps, emphasizing the order of activities. Aljahdali et al. (2012) would, although without further explanation, divide the test management into four steps: test design, test planning, test execution, and managing test defects; no other definition for test management is given. Eldh et al. (2010) divides test management process simply into two phases: test preparation and test execution, although possibly only from test management tool perspective. Davis (2006) discusses that the phases of test management can be divided into organization of test artifacts and resources, planning of testing activities as in why, what, where and when to test, authoring tests as in providing information about the how the tests are done, test execution that includes running the tests, and test reporting. Each of the literature sources has in common the need of preparation for testing or planning of it and executing the actual testing tasks according to those plans. Aljahdali et al. (2012) and Davis (2006) would include also reporting activities after the test execution in their descriptions, similarly to many other authors describing test management by activities. Interestingly, it can be noted that these steps imply that test management process includes test execution, while the numerous authors in the previous paragraph do not include it as a test management task. Having this notion in mind, the phases can be interpreted so that the actual test execution is not part of test management, but managing it may be. For instance, executing tests is not part of test management, but instead allocating resources for different tests and monitoring how it progresses are.

It has been noted that the activities in test management regards managing different elements in testing: testing process, testing data, test organization and resources (Safana and Ibrahim 2010; Liu and Robson 1992), although no specific activities or further clarifications were given. These elements do seem to characterize the activities that are considered part of test management, though. Testing process management may include all the tasks done to organize how testing is done, such as what kind of phases or levels it has, and how it is monitored and improved. Test data management can consider managing all the data involved in the testing process, such as documentation and measurements. Test organization and resource management may regard all the activities done to

resource testing effort as well as leading the testing team. Activities may also revolve around multiple elements. For instance, monitoring activities may be regarded as testing process activity by using testing data.

Finally, test management could naturally be seen only as the tasks and activities done by test manager. This may be taken for granted, as literature does not always include clear definition. However, there seems to be different roles regarding the activities regarded to be part of test management. Craig and Jaskiel (2002, pp. 19-20), although not using the term test management, would introduce four different roles in testing process: manager, analyst, technician and reviewer. Test management activities discussed in previous paragraphs, such as planning and monitoring the testing strategies and tasks, belong to test manager role. However, the tasks of test analyst, such as detailed planning and inventorying test data, may as well be considered being part of test management (e.g. Louridas (2011) and Liu and Robson (1992)). This makes the test management a discipline in which the activities are not made only by one person, the test manager, but different employees, as well. Furthermore, Craig and Jaskiel (2002, pp. 19-20) also note that it is not required that the roles are filled by different individuals.

With the literature sources introduced in the previous paragraphs, the concept of test management can be formulated for this study. Although the definitions of the term test management vary a bit in the reviewed literature, are expressed differently, or the definitions for it are not given clearly, there are clear common themes that emerge regarding what the term test management may regard. At its broadest, the software testing management tries to facilitate the testing activities as efficiently as possible through whole testing process, i.e. from the designing of test effort and executing the testing tasks to reporting the results. This includes various activities regarding managing different aspects of testing in the organization. In this study, these activities are characterized with terms test preparation, activities during test execution, and reporting results. It should be noted that these activities follow similar phases with which some authors (e.g. Abran et al. (2004)) describe testing in general. The main difference seems to be that test management does not include any activities to actually execute testing tasks, it being more of a management discipline. Test management activities may be carried out by different roles that may or may not be filled by different people.

Another way to describe the aspects may be done by using the elements in organizational aspects of testing. Using the terms from previous paragraphs, they can be defined as the **test process management**, which defines how the testing is done and organized, **test asset management**, which includes test environment, i.e. where the testing is done and what kind of artifacts that are used in testing and that need to be managed, **test team and resource management**, as in allocating the available resources to different tasks and being a leader to the testers. These elements, in fact, have apparent collections to all the elements in organization aspects specified in the previous paragraphs. The

activities, elements, and connections to other organizational aspects are visualized in Figure 5. In summary, it can be seen that test management regards managing all organizational aspects of testing. It should be noted that the concept of test management still seems to be vague in the literature, as there are numerous sources that share no clear common terminology or vision what test management is.

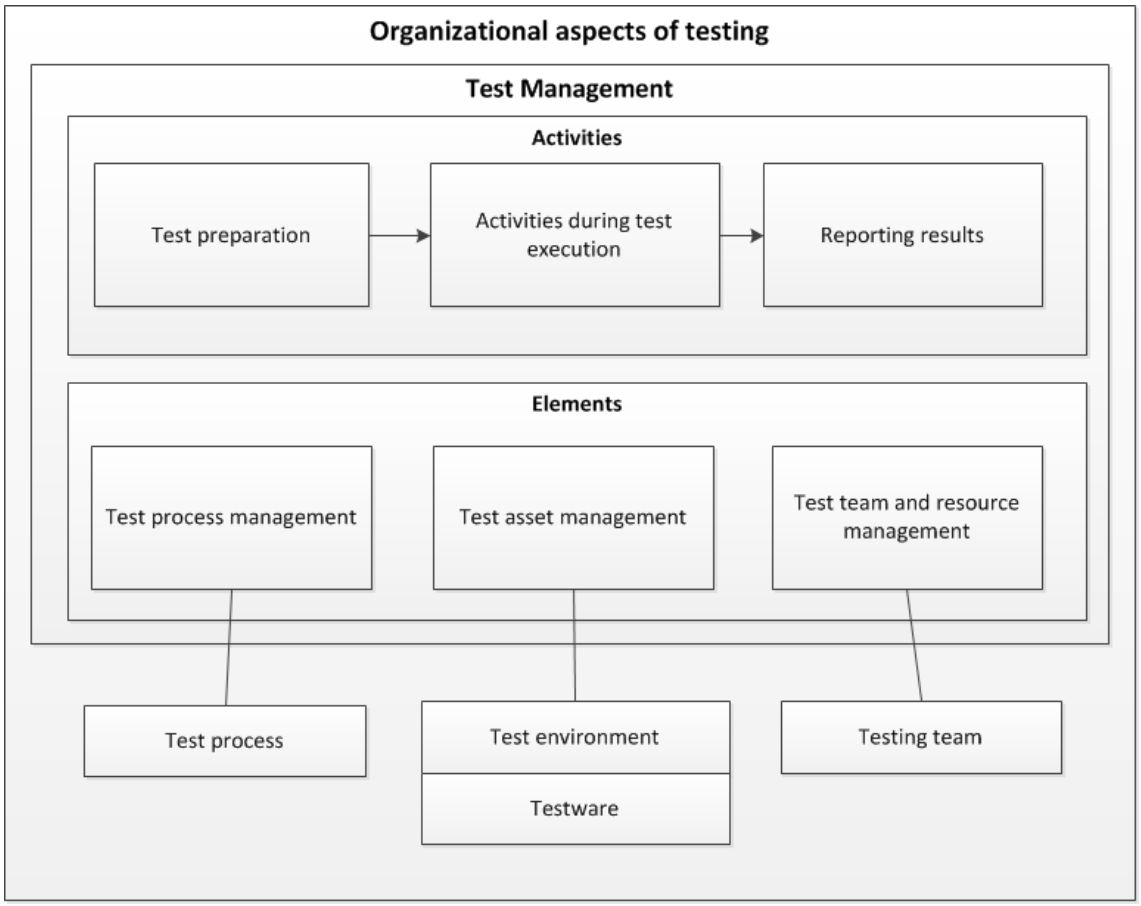


Figure 5: Test management with different aspects and phases.

The description above is not without limitations, as there are many literature sources that name these management aspects of testing differently. Arguably managing testing process may include all the other aspects, as some could include all the proposed tasks in it (e.g. Black 2002, p. 438), making managing test process and test management synonyms. However, differentiating the testing process from the other aspects of management might make the different parts of test management clearer. There are examples in the literature that would separate managing testing process from other aspects of test management (e.g. Liu and Robson 1992). Other limitation with the provided definition may be the literature explaining it; the literature with a specific definition for test management very often revolves around a tool for test management, as well (Aljahdali et al. 2012; Eldth 2010; Parveen et al. 2007; Davis 2006). In fact, none of the sources used in this study discuss test management as discipline separately without any notion of tool support. Whether or not the test management is a term coined to describe the need for

the tool, it can be argued that the importance of tasks in it is validated by other testing related literature not using the term test management.

3.2.2 Exploring activities in test management

In practical sense, it is perhaps more important to know how managing testing is done, that is, what kind of tasks it has and what is needed to do those tasks, than defining the term or concept test management. Additionally it is important to know why these activities are required. With the definition specified in the previous section, we can use a wide range of literature to help to understand how the most important tasks in test management are done without the literature sources explicitly using the term test management. This is required as many do not use the same terminology.

The following paragraphs are structured similarly to the phases of test management: preparing the testing effort, activities during test execution, and handling results. As this study primarily leads to the subject of choosing a test management tool, test asset management as well as test team and resource management aspects are emphasized. The summary of tasks is visualized in Figure 6. It should be noted that reviewed literature recognizes testing artifacts and measurements being central instruments executing test management activities. In fact, all the activities use or produce artifacts and measurements in some manner. However, the literature discussing these activities does not very often explicitly describe how these measurements should be gathered or where the artifacts should be housed, noting only that these may be needed. As they have a great impact on how the test activities are done in an organization, facilitating their gathering and use by tool support or process may be justified, though. The most notable artifacts are testing documents and test case artifacts.

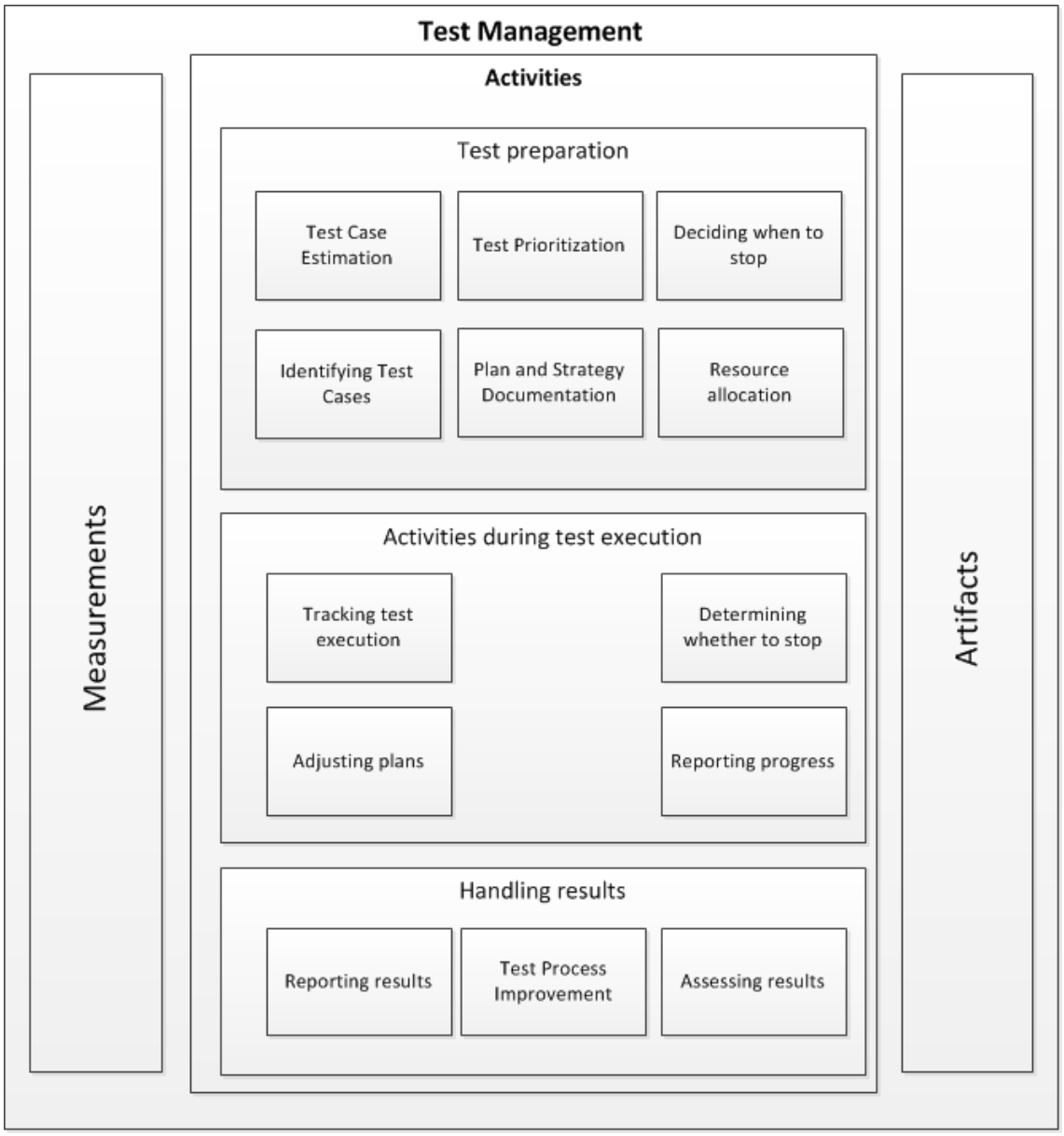


Figure 6: Breakdown of tasks in test management and central elements from test asset and test resource management perspective.

Firstly, testing documentation is seen important in the testing process. Majchrzak (2010a), for instance, notes that to a prerequisite for controlling testing is well done documentation. Whether it is part of test management, there are different perspectives. Majchrzak (2012, p. 42-43) differentiates test documenting and test management as different activities, while in his guide to ISTQB certification, Black (2008) includes documentation as a part of test management. However, as some authors have recognized designing or planning test effort part of test management (Aljahdali et al. 2012; Davis 2006), documentation of these activities may be regarded part of it, as well. It seems that documentation is central at the beginning of test execution, and thus documentation is introduced among test preparation.

In addition to documentation, another central artifact in various testing activities is a test case. The main purpose of a test case is to act as a unit, or an atomic element (Ramler 2004), for test scheduling, test execution, progress monitoring and control (Almog and Heart 2009). The importance of test case is materialized by Black (2008, p. 83), who describes test cases to be the bricks and mortar of the test system; everything in the system is designed to make the test case execution possible. Metrics, for instance, which are introduced in the following paragraphs, rely heavily on the fact that information about these test cases and execution of them are recorded. Furthermore, test cases should be traceable to customer requirements (Agarwal et al. 2010, p. 34). Test set (Craig and Jaskiel 2002, p. 229) or test suite (Eldth et al. 2010) contains a group of test cases, which cover some logical entity, such as a feature or a system (Craig and Jaskiel 2002, p. 229). As larger projects may have several hundred or thousand test cases, these test suites are the entities aggregating information about the test cases (Ramler 2004). The importance of test cases in test management is further validated by the definitions for test management in the reviewed literature; some authors would regard managing test cases, i.e. finding, designing, and assigning them to testers, the main activities of test management in their definitions (Louridas 2011; Parveen et al. 2007).

The literature often suggests the use of metrics for many test management activities, as gathering metrics from the testing process can be seen as a fundamental help to manage it. Craig and Jaskiel (2002, pp. 369; 265-267), for instance, note that metrics help in various tasks, such as decision-making, estimating schedules, tracking progress and improving testing process. They add that deciding when to stop testing may be done with defect-relating metrics. Chen et al. (2004) discuss that gathering various cost-effective metrics is important in designing and evaluating test strategy, and a "-- required competence for an effective software test manager." Coverage is noted to be most powerful metric for measuring test effectiveness. Coverage may be measured by how many test cases requirement is covered by. (Craig and Jaskiel 2002, p. 285) It is also noted that gathering measurements from testing process is essential to continuously assess quality especially in large and complex industrial software products (Eldh et al. 2010). Craig and Jaskiel (2002, p. 369) note that there is a danger of misinterpretation with metrics. Additionally as not all test cases are equally long, the amount of test cases done does not tell how the progress is. Although the metrics by themselves may show relevant information during test execution, some metrics need visualization to be understood well. Constructing reports during test execution is a way to do this. Reports aggregate test information on "different detail levels in a comprehensible and reproducible manner –" (Illes et al. 2005). The reporting may be done by exporting test results to a spreadsheet and forming charts (Farrell-Vinay 2008, p. 162).

In this study, all the activities that are done before the actual test execution are grouped under test preparation. As discovered in previous section, many authors would regard planning and designing the test effort as a part of test management. Planning is done to

ensure that testing delivers the required outcomes in time and budget (Jenkins 2008, p. 20), "—to get ready and organized for test execution --" (Naik and Tripathy 2008, p. 21) and to avoid wasteful testing as well as unproductive and unplanned testing techniques that may lead to poor-quality software (Burnstein 2003, p. 32). These plans should also be done much earlier before the actual test execution begins (Agarwal et al. 2010, p. 34). In other words, planning may be regarded activity to maximize the chance to make testing both effective and efficient. Indeed, test planning has been recognized to be important for effectiveness and efficiency (Copeland 2004, p. 212), as important as project plan for software project and as "one of the keys to successful software testing" (Craig and Jaskiel 2002, p. 54), although with no definition what is successful. However, the nature of testing makes planning challenging, as the planner needs to select certain test cases from nearly infinite number of possibilities. This is done because testing requires tight focus (Black 2008, p. 1). Test planning in itself is a major topic in testing literature, but in this study it is more important to see what kind of tasks it would include, what it needs and what artifacts it produces.

Documentation is especially important in test preparation phase. In their book, Craig and Jaskiel (2002, pp. 54-55) suggest that the testing should be well documented in many levels of detail. Master test plan covers many elements on high level, such as what is tested, how the testing is scheduled and monitored, who are testing, and what is delivered in the testing process. Similarly many use the term test strategy when describing what test planning includes on high level. Naik and Tripathy (2008, pp. 370-377) would describe test strategy formulation as determining what cycles, in each of which the quality of the product gets better, the testing process has. Detailed test planning on the other hand describes how the testing is done in different testing levels, such as in acceptance, system, or integration testing (Craig and Jaskiel 2002, pp. 54-55). Similar documents are described in IEEE standard 829-2008, which provides outline what test planning document should contain (IEEE Computer Society 2008). Producing and having documentation is important for various reasons. By using detailed test plans, the documentation of testing efforts may be divided into more effective parts (Crag and Jaskiel 2002, p. 98). These documents will also help to collect the thought and ideas of the planner and help communication with the team and other colleagues (Black 2008, pp. 45-46). Because of these transparency and communication benefits, it can be argued that having all these documents available for interested parties is important.

A part of planning is identifying test cases and determining which test tasks are more important than the others, as not all tests can be always executed (Jenkins 2008, p. 22). In regression testing, in which the amount of test cases is numerous, this may be regarded very important, and various techniques to reduce the number of executed test cases are developed. The straightforward approach is called retest-all, in which all the existing test cases in the test suite are executed. Test suites tend to grow as the software evolves, and execution of all test cases may become very expensive. (Yoo and Harman 2007)

Because of this, literature suggests different approaches to make regression testing more efficient. Yoo and Harman (2007) note that there are three approaches, which may aid reducing the number of test cases in test plan. Test suite minimization is a process in which redundant or obsolete test cases are removed from the test suite. The authors provide heuristics that require metrics and traceability between test cases and requirements for test suite minimization. Test case selection in hand tries to select the most relevant cases, that is, ones that are affected by the change in regression testing, to be executed. The third approach, test case prioritization, aims to create an order for the test cases that would maximize desirable properties. (Yoo and Harman 2007) It is noted that regression test techniques reduce the cost of regression testing, as the number of tested cases are reduced (Rothermel et al. 2001). Therefore it can be seen that by using these techniques in test management, test execution phase may become more efficient. However, these techniques require large amount of information about the program, modified version, and test suite (Rothermel et al. 2001).

The literature often describes estimating the testing effort and scheduling the testing as activities in test planning. To do this, the execution time effort of selected test cases should be estimated (Jenkins 2008, p. 23). This is fundamental to produce a schedule for the test execution (Naik and Tripathy 2008, p. 377). Effort can be estimated by using different information regarding test cases. According to Naik and Tripathy (2008, p. 377), the key information for estimation are number of test cases to be designed, effort required to create a test case, execute it, and analyzing results. They also note that there might be additional effort needed to create test environments and training and availability of test engineers. These estimations can be done by using information about previous projects (Farrell-Vinay 2008, pp. 354-360) or various techniques by using current project information (Naik and Tripathy 2008, p. 377). Both alternatives require information about the testing process, such as measurements of how many test cases there are and how long the existing test cases took to execute.

When the tests are being executed, many authors see tracking the progress of test execution a very important activity in managing testing (Naik and Tripathy 2008, p. 408; Dustin 2003, p. 300). Craig and Jaskiel (2002, p. 258) note that “one of the first issues that a test manager must deal with during test execution is finding a way to keep track of the testing status”. Similarly Dustin (2003, p. 200) states “everyone involved in a software project wants to know when testing is completed --”. This indicates that the progress needs to be communicated between parties. Monitoring test status may be done various ways. The current status of testing can be simply communicated between the tester and test manager via e-mail or by conference call (Black 2002, p. 395). However, very often the literature groups metrics gathering and progress monitoring together (Black 2008; Farrell-Vinay 2008, p. 110; Dustin 2003, p. 200), making the former an instrument for the latter. Jenkins (2008, p. 27) states that without advanced metrics, one can only measure the amount of time one has left and adjust the plans according to that.

Other ways to report testing status is to use the milestones completed, coverage achieved, as well as number, severity and location of defects discovered (Craig and Jaskiel 2002, p. 258). Naik and Tripathy (2008, pp. 408, 419) would divide metrics into two categories: the status of system testing and the status of defects. These metrics may be used by the management team to make pre-emptive and corrective actions. Finally, as many criteria for stopping testing may involve metrics, such as meeting testing coverage goals or defect discovery rate has lowered to certain threshold (Copeland 2004, pp. 237-238), metrics may be regarded essential for getting information about when to stop.

After test execution is done, the results need to be reported and communicated in some manner. The topic of summarizing or reporting results after test execution is widely recognized in different literature sources (Farrell-Vinay 2008, p. 156; Burnstein 2003, pp. 221-224; Craig and Jaskiel 2002, pp. 258-260). One often described report is test summary report, which gathers all the results of the testing activities. The main purpose of these reports is to provide advice on the release readiness of the tested product, identify limitations of the software and failure likelihood (Craig and Jaskiel 2002, pp. 258-260) as well as provide a basis for lessons learned for future projects and help to evaluate the effectiveness of the testing team (Burnstein 2003, pp. 221-224). Hence it can be argued that in order to deliver the developed product to a customer, the results of the testing need first to be analyzed. Reporting results requires a great amount of information from various sources, as it summarizes results, activities and events, including defects, resource consumption, task durations, tool usage, variances between original test plans, procedures, and designs and how they actually ended up (Burnstein 2003, pp. 224). Arguably having this information as accurate as possible may increase the correctness of the reports, and subsequently the basis for improvement. The reviewed test management literature does not explicitly say how this information is gathered, only what it should contain. Having accurate information may require measurements by automatic means or manually gathering the data.

3.2.3 Importance of test management

The benefits of separate activities regarding test management are widely recognized in the literature and included in the previous section. The importance of test management in software engineering context, however, is not very often explicitly expressed in the reviewed literature. In various sources, Majchrzak (2012; 2010a; 2010b) discusses the benefits of test management. From one perspective, Majchrzak (2012, p. 44) notes that there is always a need for managing one’s own testing effort. In organization context, managing of the testing effort of everyone may become more relevant. Eldh et al. (2010) note that because of complexity of testing activities, efficient and scalable test management is required in an organization. With this statement they may be referring also the tool support for test management. Majchrzak (2010a) suggests that when the number of employees increases in the organization, the company should "install a de-

partment that will coordinate testing companywide", which could provide guidelines and produce testing process, i.e. to perform test management activities. Test management can be seen important particularly when the testing is done in teams, a software developing company employs dozens of employees or a controlled testing process is emphasized. It can be seen to contribute to improved overall software development process, while decreasing costs and increasing software quality (Majchrzak 2012, p. 43).

According to Davis (2006), the "-- general goal of test management is to allow teams to plan, develop, execute and assess all testing activities within the overall software development effort". In other words, test management can be seen as a set of activities to support all testing tasks in software development context, giving it a supportive role. Another aspect speaking for test management is improved effectiveness in testing activities. Parveen et al. (2007) note that "one way to support quality assurance is efficient software testing --" and state that organizing test artifacts is one of the many ways that would make the process more efficient. Also, as many stakeholders look into ways of reducing the cost of testing, test engineers need to select fewer, the most effective test cases for execution (Naik and Tripathy 2008, p. 11). This could be considered to be part of planning activities of test management, in which test cases are selected to be executed. Majchrzak (2010a) notes that test controlling is important as it can identify problems in testing performance. This could naturally lead to fixing these problems and enhancing performance. As a closely related topic, literature also discusses the need for measurement in software development activities. According to Chen et al. (2004), an effective "measurement processes help software engineers succeed by enabling them to understand their capabilities, so that they can develop achievable plans for producing and delivering products and services of required quality". In a sense, measurements and metrics will help in planning and estimating needs of testing.

The reviewed literature introduces systematic ways to execute testing activities. Furthermore, with the literature review before, it may be argued that dedicated test management will make the management more visible and effective, as documentation is created systematically and the testing planned and monitored so that it is effective. The activities and elements of test management described in this section may be regarded as a set of requirements for a tool; these activities need to be facilitated by the tool. As a summary, it may be argued by reviewing literature that test management makes the testing effort more systematic, controllable, and more capable to improve. The need becomes greater when the system under testing becomes more complex and more people are needed to test it.

3.3 Challenges and difficulties in managing testing

As concluded in the previous section, test management contains valuable set of activities for software engineering discipline dealing with many elements in it, such as testing

process or tool support. However, like any other business process, it has challenges and difficulties executing these activities. This section will lay basis on difficulties by describing what problems the literature has acknowledged. A summary of difficulties in testing is characterized by Craig and Jaskiel (2002, p. 9), who state that "-- testers may have different reasons why they think testing is difficult, but they all seem to agree that IT IS DIFFICULT!" Although they may be referring to test execution activities, many problems are related to management of testing, as well. Many of the problems seem to be commonly known and experienced by different authors. However, there does not seem to be much research or empirical evidence regarding difficulties specifically in test management context. The literature used in this section hence covers mostly the challenges in software testing in broader context, and the management aspects need to be connected by other means. Closest to the problems from management perspective is a study by Lyles (2013), which includes interviews of expert test managers. On broader context on testing difficulties, only the research by Kasurinen et al. (2009) was done as a multiple-case study involving large and small companies from various business segments. Otherwise, the literature sources mostly note problems without really showing any evidence of them. There are clear common themes that emerge from the literature: the inefficient testing processes, test planning problems, test asset management, and lack of tool support. Figure 7 visualizes the challenges linked to test management activities and elements, giving context where the problems affect.

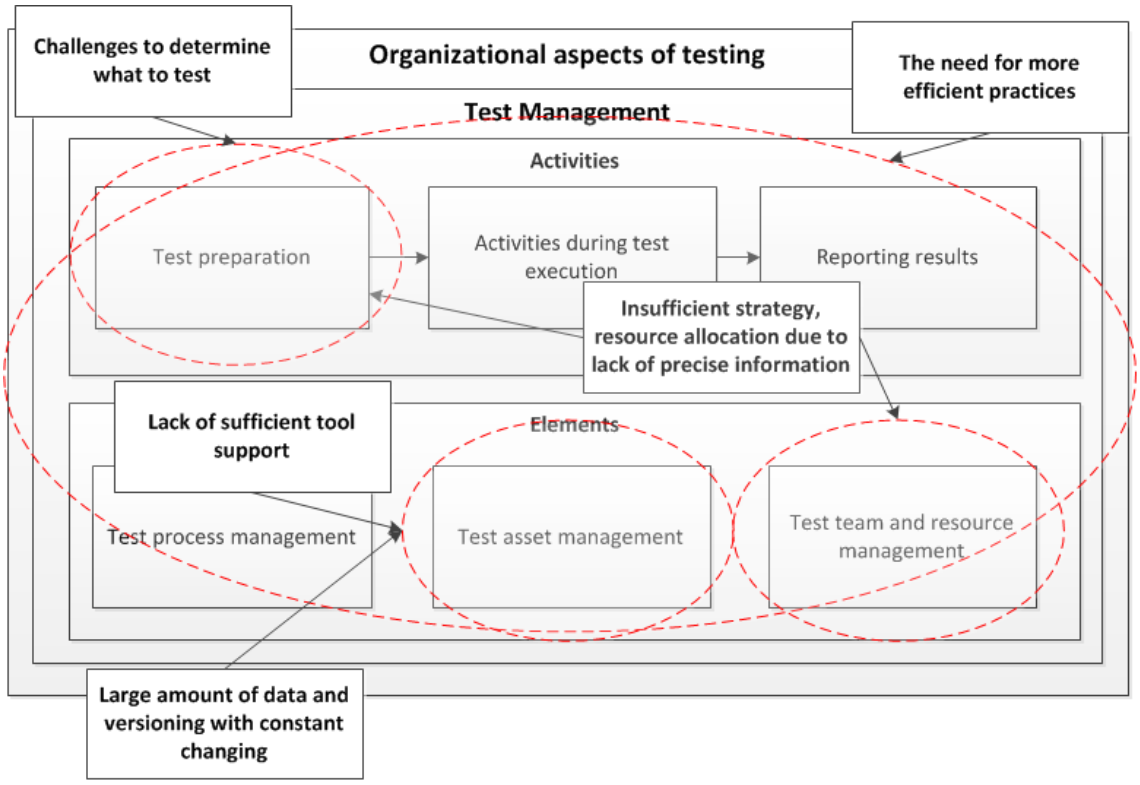


Figure 7: Challenges in managing testing linked to activities and elements in it.

There are various literature sources explaining difficulties in software testing, which can be linked to the need for more efficient testing practices. The basic need for improving

efficiency is an old basic software problem called software crisis. Even today, it is acknowledged that it is regular that software projects "-- are not only delayed, but exceed the allowed budget by far due to software problems --", or fail altogether because of these problems (Majchrzak 2010a). Badly executed or ineffective software testing is not the only reason for these problems, but as testing can be seen as main way to assure quality, increasing its effectiveness may be regarded important. Often the literature also highlights the competitive environment and the status of testing within overall development cycle. Software developing organizations are in problematic situation, as they aim to shorten the development time and improve the quality of their products (Lazic and Mastorakis 2008), while economic situations may prevent organizations from making required investments to assure quality of their products (Ramler 2004). Andersin (2004) adds while organizations aim to shorten the time-to-market for developed products, there may be "--no evidence of decreasing the number of errors made in a certain period of time". Therefore it can be argued that current practices are not inefficient, but for the competitive environment it would need to be more efficient.

Many problems in software quality may be due to not making the necessary investments on testing activities. This is widely accepted in the reviewed literature, but it is not backed up by empirical evidence. In software companies, it has been noted that testing resources and time are usually limited (Parveen et al. 2007; Craig and Jaskiel 2002, p. 9), and the project managers may need to choose whether to meet the deadlines or do comprehensive testing (Ramler 2004). On the other hand, in order to meet deadlines, the employees have to work overtime or the management cuts the testing effort (Kasurinen et al. 2009). It has been recognized that the staff does not get enough time to test the developed product properly, thoroughly, or well enough (Copeland 2004, p. 4; Andersin 2004). Although the reason for this may lie in top management for releasing untested product to customers, test management may be faulty of not planning, allocation, or improving the processes appropriately. The lack of time for testing has also been noted to cause other problems, such as conflicts between testers and developers (Cohen et al. 2004). Partly related to insufficient investment, resourcing testing correctly has also been noted to be a problem in the literature. Kasurinen et al. (2009) write that testing should be executed by dedicated testing personnel, but smaller companies may not have the resources to do that. The lack of resources may put pressure on testing to be more efficient, which again may require good test management practices.

Managing testing as a discipline has many problems in practical settings. Andersin (2004) notes that testing is not managed well, it "-- takes too much time, costs more than planned, and offers insufficient insight of the quality of the test process". While this again suggests process improvement approaches to increase efficiency, improving testing practices is not an easy task, either. Problematic in executing and improving testing practices is that there is no universal testing theory developed (Bertolino 2007). Instead, software testing practices are communicated through best practices and standards

(Taipale and Smolander 2006), but on the other hand there is no set of best practices, which is generally known and that could be implemented in any company (Majchrzak 2010a). The best practices in one company may not therefore be transferred as such to other company. Hence, the best practices for a company may need to be developed inside the organization or customized to fit for the certain processes in it. Finally, because of these difficulties organizations have started to introduce testing earlier in processes or trying to find more effective testing strategies, but only small portion of them have established any real means to measure effectiveness. Thus it is argued that organizations are not really able to improve their effectiveness. (Parveen et al. 2007)

Test preparation phase, during which all the planning is done, has been noted challenging. Software testing itself has been noted to be very complex process and becoming more complicated as the systems under test become larger and more complex (Taipale and Smolander 2006). The research regarding challenges in test efficiency is also linked to the always increasing complexity of system under test (Bertolino 2007). Therefore planning testing activities may be seen problematic, as well. Firstly, Andersin (2004) states that some organizations do not introduce testing in right place in software development. Kasurinen et al. (2009) observed that software companies need better testing strategy or planning to allocate resources better. Similarly a test manager interviewed by Lyles (2013) notes that team member allocation to different tasks is a challenge. Liu and Robson (1992) discuss that early planning is difficult due to lack of precise information, which makes the allocation of resources problematic. Difficulties in planning may cause problems afterwards. Safana and Ibrahim (2010) note that difficulties in early planning cause many problems to other test management activities. They also add that early planning is key factor to success.

Another central issue in managing testing is that there are too many combinations of inputs to test (Copeland 2004, p. 4) and not all possible combinations and settings can be covered (Naik and Tripathy 2008, p. 13). Even if it was possible to test everything, one might argue that it is not feasible. Ramler (2004) argues that focusing on the most valuable tests is the key to effective software testing. It is noted that before executing test cases it is difficult to know which ones are the most valuable, but in situations where the product is tested multiple times with the same test cases the value of each case can be done more easily. For this, one needs to record information about defects found with the specific test cases, and may in practical sense need an automated support for decision-making. (Ramler 2004) In her summary of current challenges in software testing, Bertolino (2007) finds effective regression testing being still a challenge because of the dynamic evolution of system under test; it is not known what to test and how often. Furthermore, due to expensive nature of regression testing, test selection techniques should be effective and cost-efficient. Regression test selection techniques are numerous, as noted by Engström et al. (2010), but the variety of techniques suggests

that there is no general solution for it. They also conclude that there is no strong empirical evidence of picking one test selection technique over others.

Test asset management is seen problematic for organizations in various literary sources. Based on the observations made in their case study, Safana and Ibrahim (2010) state that due to large amount of data and versioning, test management is difficult. It has been noted that organizations do not utilize test cases created in earlier stages of testing, which makes testing ineffective (Majchrzak 2010b). Root cause of some of these problems may be the continuously changing software and requirements. Lyles (2013) quotes a test manager saying that test managers create order from chaos, they "sit in the very middle of CHANGE", and that effective communication is required. Also Liu and Robson (1992) note that as the testing process is about checking the developed item against its specifications, which are always changing, the current version must be known at all times. They also note that the relationships between different data are complex and likely undocumented, making maintaining the relations difficult. This makes regression testing ineffective, and as the test cases and links between those and requirements are poorly maintained, testers may not know whether the witnessed functionalities are by design or not. Furthermore, it is difficult to know which cases should be selected to be rerun as there may not be a way of knowing which test cases the implemented change has affected. Indeed, it has been noted that the problem of conflicting visions regarding what the software should do is a challenge that test managers need to face (Lyles 2013). Some of the problems are not regression testing specific, either. In more general context, Copeland (2004, p. 4) notes that it is problematic to determine the expected results of each test and lists nonexistent and rapidly changing requirements as a difficulty. Test case integrity is also been noted to be a problem. Desai (1994) has noted that before implementing test case management tool, testers had different means of writing test cases and execution results. Without a tool, it was very tedious to compare test result histories.

Finally, lack of sufficient tool support may be regarded an issue in test management. Copeland (2004, p. 4) note that one of the problems in testing is lack of tool support. Even though using testing tools has been noted of being very important, it is also noted that having tools is not a solution to all problems, either, as even the automated test cases cannot be run every day (Eldh et al. 2010). Safana and Ibrahim (2010) also identify difficulties that emerge when not using a tool to help some aspects of test management tasks. They recognize that processes are ad-hoc and not repeatable, the connections between test cases are not clear, requirements and defects cannot be visualized, monitoring progress and productivity during testing is difficult, sharing information and getting real-time metrics is hard and there may not be central repository of test results. Kasurinen et al. (2009) report that faulty or complicated testing tools cause additional resource losses and that the tools may restrict their own testing processes. Implementing a system to acquire metrics in testing process is not problem-free, either. Chen et al.

(2004) note that as testing does not have elements of physical quantity, measurements in software development and test process is complicated. They also noticed problems in acquiring metrics from testing process in their pilot project. They had tools ready to acquire the necessary data, but in practice they needed to redefine the tools to acquire them. Other problems noted when implementing testing tools are high costs and the fact that they are hard and slow to use (Ng et al. 2004). On the other hand, it is noted that tools in testing are often used in organizations, but they are not integrated to other systems (Majchrzak 2010b), which increases the need to manually handle information and therefore several activities may become unnecessarily time-consuming (Eldh et al 2010).

3.4 Test management tool

Generally, as a tool may be defined as being a device or implement that is used to help perform a job (Oxford Dictionaries 2013), a test management tool could simply be defined to be a software or hardware solution that helps to perform test management activities. Hence the definition for test management provided in the section 3.2 is extremely valuable; to define what test management tool is, we need a definition for test management in order to know what the tool is for. However, test management tool is regarded various ways in the literature, and different implementations of it concentrate on certain aspects of managing testing. This section will lay basis on this central subject in this study, such as what test management tool is and what it should do. Furthermore, it is discussed what kind of test management-related difficulties a tool may help or even solve. Investigating these aspects provides information regarding various requirements and considerations that there may be from the literature point of view. First, the concept of test management tool is introduced, after which different considerations of the tool are investigated. Again, the literature used in this section consists mostly of books and publications that are not based on empirical evidence but rather on personal experience in single environments. This may be regarded an acceptable limitation, as the collection of personal experiences may be used to form sufficient understanding of the interesting subjects for this study.

3.4.1 Concept of test management tool

According to Craig and Jaskiel (2002, p. 216, 448), a "-- testing tool is a software application that helps automate some part of the testing process that would otherwise be performed manually", and it might be software or hardware product. Although a testing tool automates or makes executing any testing activity easier, the term automation in testing refers usually executing test cases automatically. While reviewing software testing literature regarding the topic testing tools, one may observe that it focuses on automated test execution (e.g. Naik and Tripathy 2011, p. 24). It is important to differentiate these tools when reviewing literature. A tool for test management, on the other hand, is not often discussed topic in the literature. This has also been noted by Eldh et al. (2010),

who state that the tool for test management "-- is surprisingly scarce in research papers, as vivid as it is in usage --". This makes it difficult to gain reliable information what features test management tool should have.

Test management tool may be considered part of a wide range of maintenance tools that may help the "quality of a modified information system and the productivity of the system maintenance process" (Khan et al. 1997). The literature seems to use different names for a tool related to test management based mostly on their main functionalities, for example test case management tool to house test cases, test execution data, and test reports (Louridas 2011; Majchrzak 2010b; Desai 1994), a support environment that adds links between test data to increase traceability and help managing the changes in different elements (Liu and Robson 1992), and regression tool to help managers to choose which test suites should be selected for regression testing (White et al. 1993). The first two tools concentrate more on test asset management aspects of test management, while the regression tool also include resource planning aspect.

Test management tools seem to include many of the functionalities for test asset and resource management (e.g. Black 2008), as well as other features that make managing testing easier. According to ISTQB glossary, a test management tool is a "tool that provides support to the test management and control --" (Black 2008). Therefore a test management tool, in a sense, might help any part of the test management process, being it test preparation, activities during test execution, or handling test results. In fact, as test management process includes many roles, it is noted that test management tools are used by test managers and test analysts (Black 2008). According to Eldh et al. (2010), a test management system may even diminish the need for a test manager in test execution phase, as test case correction monitoring and reassigning tasks can be done automatically. There seems not to be only one way to implement a test management tool, but instead different approaches have emerged.

At its simplest, a test management tool can be pen and paper, a word processor or a spreadsheet (Davis, 2006). Usually the starting point of test documentation is indeed done by using spreadsheets (Majchrzak 2012, p. 43). Examples of spreadsheet implementation of a test management tool are introduced by Black (2002, pp. 179-212) and Farrell-Vinay (2008, p. 142-146). In his book, Black (2002) presents a system to track which test cases are run, how many have failed and by whom. It is also possible to extract metrics from the spreadsheet. Farrell-Vinay (2008) similarly monitors the testing efforts with a spreadsheet system, to which status of testing and defects as well as log entries are inserted manually to get simple reports and trends from the test process. Additionally, word processors, spreadsheets and pen and paper solutions in test management may be cost efficient and easily built, while building a larger database-based tool may be laborious to develop (Black 2002, p. 209). However, using these techniques may be rather manual and lose their effectiveness when real-time monitoring is needed

or when test cases are numerous. Black (2002, p. 209), for example, notes that in order for his spreadsheet to work and using it not become too laborious, a single test case should take hours to be executed. The monitoring spreadsheet by Farrell-Vinay (2008, p. 142-146) is also mainly maintained by manually inserting information or updating the information by automated means at the end of a certain time period.

Compared to the traditional techniques to monitor the progress of test process, more sophisticated tools for test management are present in the literature. When the testing effort becomes larger, an organization may use internally developed test management solution that is based on spreadsheets or databases, or the organization may introduce a commercial system for their test management needs (Davis, 2006). The different implementations of test management tool may be done sequentially as the company needs them. A simple version of test management tool may be introduced first to the company, and the further improvements may be done later (Majchrzak 2010a). The need for more sophisticated implementation is needed when company grows larger. Majchrzak (2010a) recommends that larger software enterprises should adopt this kind of system, which can also be connected to other tools of testing. The literature does not often identify specifically the level of test management system needed to successfully manage testing. Few authors, especially those who introduce new test management systems, do discuss the benefits of high level of automation (Eldh et al. 2010; Safana and Ibrahim 2010), but many do not specifically address this, leaving the definition of system for the readers (e.g. Majchrzak 2012; Louridas 2011). As Majchrzak (2012) states, testing tools makes testing more effective, "-- relieve humans of repetitive tasks and make testing more economic –", it could be argued that further automation would make performing various tasks easier.

Literature recommends the use of some kind of test management tool. In general, it noted that it "-- is generally accepted that appropriate maintenance tools can have significant impacts in assuring the quality of a modified information system --" and productivity of maintenance process (Khan et al. 1997). Using a test management tool is suggested especially with regression testing. Majchrzak (2010b) notes that test cases should be stored in a tool in order to make testing in between different testing levels more efficient. Burnstein (2003, p. 496-497) states that test management tool and advanced test management systems are part of advanced test mature model levels. These systems should have "-- centralized, sharable location for all test-related items –", and provide information for test process optimization, quality control, and defect prevention. Test tools can be seen useful when they take over repetitive tasks, a high level of precision is needed, metric and documentation of testing is wanted to be automated, or there is a need to process massive amounts of data (Majchrzak 2012, p. 47). Craig and Jaskiel (2002, pp. 214-218) also recognize that only repetitive tasks, such as regression tests, and tedious tasks, such as human-intensive tasks or calculations, should be automated with some type of a tool.

Actual results of implementing or having a test management tool has not been well documented in the academic literature, and there are only few case studies that discuss this. A case study by Safana and Ibrahim (2010) involves implementation of SpiraTeam Tool in a software company and focuses on how the tool was integrated part of the testing process, describing mostly the steps of the implementation and illustrating the benefits against manual test management on conceptual level. The experimental results focused on how the tool produced various reports of testing activities. Unfortunately there was no empirical evidence presented towards the new system being any better than the old one. They, however, note that manual test management has numerous disadvantages that can be solved by using a tool. (Safana and Ibrahim 2010). Another case study by Parveen et al. (2007) describes the process of migrating from Microsoft Excel and Word worksheets to a test management tool called TestDirector. During implementation, they noticed that the tool had problems regarding converting the test cases from the old system to the new one. In the tool, the test cases could not be described the same way as they were before. Even after a year, the limitations, such as poor performance and lack of customization in test case format, remained to difficult the use of the tool, and some teams could not get the advantages the tool might have brought. Eldh et al. (2010), on the other hand, have had positive experiences with their implementation of test management tool, as by automating part of their processes, they have witnessed increase in test execution efficiency. In software product lines context, Neto et al. (2012) recognized significant increase in designed test cases, efficiency in test case design, efficiency in test case execution, and errors found when an internally developed test case tool was introduced in their process. From these experiences it can be said that having a test management tool may be regarded a positive addition to testing process, but it needs careful considerations and planning before implementation. Unfortunately the first three reviewed articles did not present their context very well, which raises questions that is having the tool as beneficial in every context. However, software product lines may be regarded close to evolutionary development, as both eventually develop and improve existing systems, not always create new software systems.

3.4.2 Considerations for a test management tool

The literature shows various common elements that should be included in a test management tool. Alternatively some articles discuss the desired features or characteristics in this type of tool. In Figure 8 is visualized the different desired features discovered in the reviewed literature linked to activities and elements regarded part of test management. It can be seen from the figure that all the activities are covered by test management tool functionality, while test process management, as in how to test, may not be regarded part of the functionality of a test management tool. This suggests that a test management tool with all the desired functionalities tries to facilitate all the activities without any implications to the test process. In addition to functional requirements, the literature sets a few quality requirements for a tool. Most notable of these are integra-

tion, customizability and extensibility, and performance and usability. These aspects are reviewed in the following paragraphs.

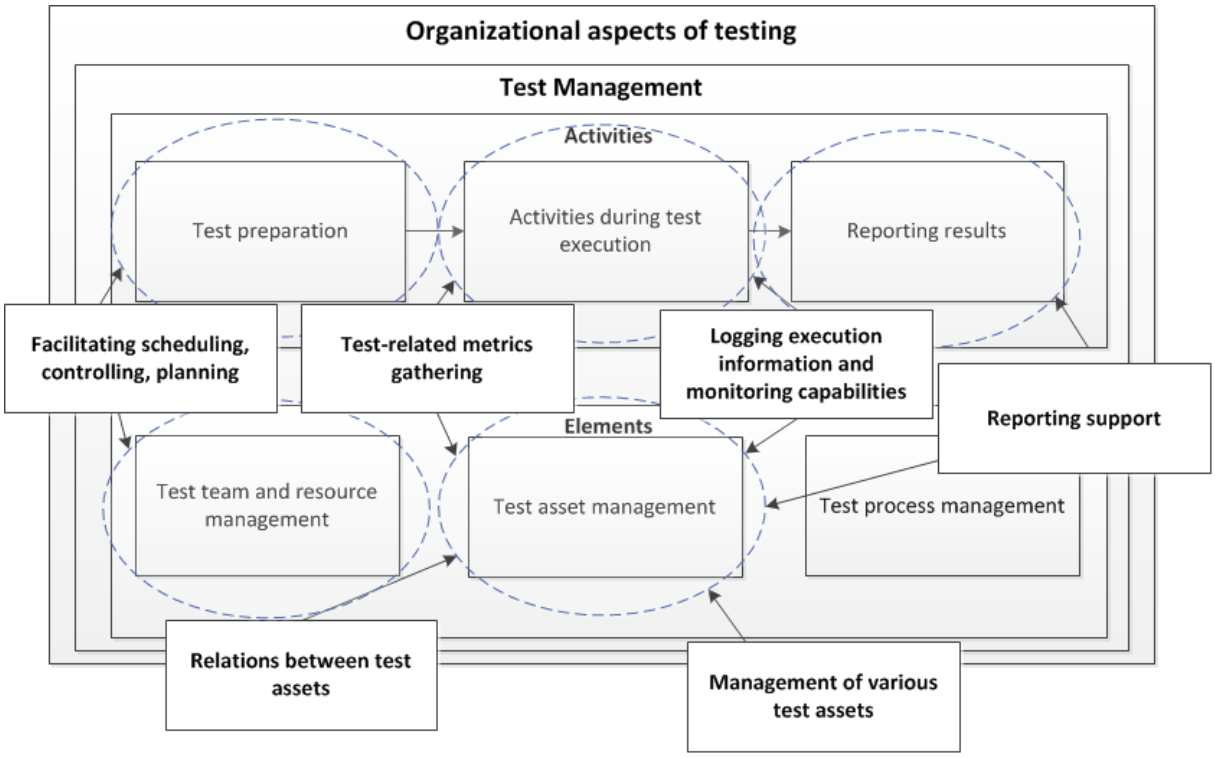


Figure 8: Commonly desired features of test management tool linked to test management activities and elements.

There are a few limitations to this set of characteristics. The methodology behind the characteristics is not always backed up by evidence that is clear to the reader. Some of the features are introduced as part of concept for a new system (Eldh et al. 2010; Neto et al. 2010), from interviews on a multiple case study (Majchrzak 2010b), based on test maturity model (TMM) (Burnstein 2003), and based on certification organization guidelines (ISTQB Guide n.d.; Black 2008). The last two, for example, may be regarded insufficient for research because of the lack of visibility where the characteristics come from. On the other hand, as testing practices are communicated through best practices, these considerations may be thought as such, and provide some value to the study. It should be noted that reviewed literature also had listed these features for different reasons in different contexts. Some of the authors listed the features as required ones for test management in large software systems (Eldh et al. 2010), some possible features of the system (ISTQB Guide n.d.), and some as their own requirements for test management tool in product lines context (Neto et al. 2012). As they were listed for different goals and contexts, it cannot be stated that these features would be required for all test management tools. Rather these categories show the different sets of features that a test management may have. Like noted by Burnstein (2003, p. 496-497) commercial versions including all the desired features for test management system may not be available. One could argue that the needs of the organization specify which of these features are relevant in each case. Rationale behind why these requirements were desired would

have been relevant for this study, but unfortunately most literature sources omitted this from the discussions.

One of the most often discussed features is test case and other test asset management. ISTQB Guide (n.d.), for instance, notes that test case management is one possible feature for a test management tool. From one perspective, test case management features the tool should include are the compilation and categorization of test cases (Majchrzak 2010b). Similarly Eldh et al. (2010) state that the tool should record information about test cases, including their names, priorities, and steps as well as version and change information regarding different test assets. In product lines context, Neto et al. (2012) note that the tool should include various features regarding test case management, such as a possibility to import, create, and store test cases manually and from file, generate test cases automatically, a way to reduce the number of test suites, enable, disable, remove, and export test cases. They also add that these test cases should be visualized in some manner in the tool. Other assets the tool should house are environment data for complicated test environment (Black 2008), test suites, test plans, and build information (Neto et al. 2012; Burnstein 2003, pp. 496-497). These assets should be reusable, i.e. the tool should allow using these assets in different situations, as well (Neto et al. 2012). Burnstein (2003, p. 496-497) states that test management system should provide a place for all testing team members to acquire information about testing. All the described functionalities have in common the need for managing test assets, while the proposed implementations may differ slightly.

It can be seen that possibility to create and visualize relations or links between test assets is noted important for a test management tool in the literature. Black (2008) sees that the tool should facilitate traceability of test artifacts and the test basis, such as requirements or risks. Burnstein (2003, p. 496-497) would also include the ability to link test cases to requirements as a feature in advanced test management system. Similarly ISTQB Guide (n.d.) would add traceability from testing assets, testing results, and defects to requirements of system under test as a feature of a test management tool. These relations may be helpful when calculating test coverage, a feature which also has been noted to be a requirement for a test management tool (Neto et al. 2012). In regression testing context, Onoma et al. (1997) note that a tool should contain information about relationships between test cases but also add that there needs to be information about relationships between changes. The need for this kind of feature may be derived from the problems of continuously changing elements in testing noted in previous section.

Planning, scheduling, and controlling test execution activities by assigning test cases to different testers are mentioned functionalities in the literature. Eldh et al. (2010) note that test scheduling and planning with time estimations and allocations of resources between test cases would be a mandatory feature for a test management tool. Also ISTQB Guide (n.d.) notes that a test management tool could include test case scheduling and

managing testing related activities. A tool may also help implementing regression testing techniques in testing planning phase (Yoo and Harman 2007). As planning in test management may be difficult, this feature could be helpful to create better plans and schedules. It can be seen, though, that the feature for test planning is not as largely desired functionality as the other ones.

As test management requires means to monitor and communicate testing status to other parties, similarly often noted functionality for a test management tool is a way to log execution information and monitor the execution in some manner (e.g. Burnstein 2003, pp. 496-497). In fact, many literature sources describe this functionality very similarly, but the implementations are explained a bit differently. In ISTQB Guide (n.d.), for instance, it is noted that a test management tool could log test result from both manual and automatic execution, and Majchrzak (2010b) notes that the tool should allow setting a state to a test case. As required functionality for a tool, Eldh et al. (2010) include the possibility to see the progress of test execution including the number of test cases run, passed, and failed. In regression testing context, Onoma et al. (1997) would also include monitoring test progress, defects found, and the actions done for these defects as functionalities of test management tool. Black (2008) notes that test management tool should help track concurrent test execution in different environments and different locations. In the end, all the implementations have in common the need to log information about testing effort and a possibility to see how it is progressing.

As there are great amount of data recorded in the system, these can be used various ways. Foremost functionality noted for information mining is the ability to create reports from the gathered data. Similarly to monitoring test progress, as the literature sources regarding test management included reporting, the same feature is wanted in test management tool context, as well. Burnstein (2003, pp. 492; 496-497) would include measurement capabilities and reports on testing status features in the test management system. Neto et al. (2012) see that generation of report of test cases is one requirement of test management tool. The tool has been noted to acquire metrics from the process to help reporting. In ISTQB Guide (n.d.) it is noted that the tool could visualize the progress with metrics and reports. Similarly Black (2008) notes that various test-related metrics should be gathered with the tool. Burnstein (2003, p. 492) sees generating reports on, for example, testing status being useful in a test management tool. Again, specific implementations of how the reports should look like are not described very extensively, and only the need of having reporting capabilities, such as producing progress and other informative reports, in the tool is desired.

The need for integration between different systems has been well noted in the reviewed literature. A test management system described by Burnstein (2003, pp. 496-497) and in ISTQB Guide (n.d.), for instance, would need to be well integrated with other tools used in software development. Full integration of tools brings new possibilities to get an

overview of the testing process and calculating key figures. Integration interfaces of different systems in a test management tool may include test case management, development or project planning, test scheduling, staff assignment, time control, task management, test controlling, and even a management cockpit (Majchrzak, 2010b). The importance of integration is also recognized by Eldh et al. (2010), who state that the main strength of their test management system was the possibility to have it interacting with other systems, such as failure handling, build management, and test automation systems. Similarly extensibility, as in adding new features afterwards, has been noted to be a non-functional requirement (Neto et al. 2012). It is understandable that extensibility and integrations are important for a tool, as it can be argued that these improves the organization's capability to automate greater part of the processes.

Test management systems may require a certain process, and an organization may fit its processes to the one of the tool. While the integrations and extensibility emphasized on the need to have the new system integrated well with the existing infrastructure, customizations seem to address the need for the tool to adapt into current and future processes. Majchrzak (2010b) argues the importance of customization lengthily. If company has a defined testing process, using a tool that tries to change that process may be considered a bad idea. Customization is important especially when the company wants continually improve their processes. For instance, customizing interface should be possible. For example steps to create test cases need to be adapted to the workflow of test designer. If the tool forces the tester to work a certain way, it may end up reducing productivity. (Majchrzak 2010b) On the other hand, some good practices may be communicated and forced by the tool and not be a bad addition to a process. This is not discussed in the examined literature, though.

Finally, apart from integration and customizability capabilities of the tool, the literature regarding test management tool does not often discuss other quality criteria of the tool very extensively. There are a few considerations that single literature sources introduce. As their requirement for a tool, additionally to reusability of testing assets, Neto et al. (2012) list performance, platform independence, and usability as quality requirements for their tool. There are many features that help the communication or the usage of the tool. For instance the tool could offer reminders and connections to the environments that are used to run the cases automatically (Majchrzak 2010b). Burnstein (2003, p. 492) notes that the user interface is valuable capability of a test management tool. It is also noted that the test management tool should be able to manage the testing process in one environment (Safana and Ibrahim 2011), and ideally an employee knows exactly what to do at any time by looking at the tool (Majchrzak 2010b). These two notions seem to point to the need to have different systems interacting with each other, so that the users do not need to use multiple tools.

3.5 Selecting software systems

Buying software systems may be linked to more general organizational buying context, which may be considered an art form in itself. Kotler and Keller (2006, pp. 210-211) note that business markets have various characteristics that distinguish it from customer markets, such as the need for multiple sales calls and multiple buying influences. In a situation in which a new product is purchased, the buyer has to make many decisions, and various influentiators participate in the process. Software companies that develop their own products or have a reasonable amount of know-how in programming have a possibility to develop a software solution for their needs using the resources the company has. The chance to develop fully customized product, for example for test management, brings another full dimension into the decision process. The central problem with the added dimension is the need to analyze when it's profitable to develop the solution internally or when a readily developed commercial product is a better choice. In addition to the increased complexity of the decision process, the companies need to know exactly what kind of tool they are looking for and what kind of effort is required. In this section, first the literature regarding choosing between developing own solution and buying readily developed product is reviewed. In the second part of this section the literature regarding how to choose from different available products is reviewed.

3.5.1 Choosing whether to develop internally or buy

The make-or-buy decision is relevant in many industries. Choosing between developing fully customized software solution and buying ready software is no trivial task. In the following paragraphs the discussion whether to develop internally or buy from external source will use the term build or buy, similarly like in the reviewed literature (Langer 2012; Ledeen 2011; Daneshgar 2011). Although there might be a possibility to outsource the software development or parts of it to an external developer to achieve customized solution while not needing the necessary resources, this possibility is not included in this work.

In Table 2 is a summary of the common themes found in the literature grouped under terms used in this study with the relevance to the reviewed literature. Different sources seem to issue the same points regarding what should be considered in decision-making. The main points discovered in the literature for this study are the ones that have to be considered when deciding whether to build or buy: how important the activity for which the desired product is wanted, how well the requirements fit to the candidate, how hard it would be to develop it in-house, how organization's current IT structure fits for the candidate, how well the organization could develop the solution, and how costly a candidate would end up. Similar discussions are included in the literature in other industries, as well. For instance, in his book regarding outsourcing McIvor (2005) notes that there are two dimensions when deciding to outsource or not: the importance of the activity and the capability of an organization. There are clear connections to the elements

in the table. The significance of the product has a clear link to the importance of the activity, while the characteristics of the organization include the internal capabilities to develop the tool. It can be argued that similar priorities exist in other industries, and there might be helpful knowledge gained from them.

Table 2: Criteria regarding build versus buy according to the reviewed literature.

Main criteria	**Reviewed literature**
Significance of the product	Core vs. Context (Ledeen 2011); Strategy and competitive advantage; Intellectual property (Daneshgar et al. 2011)
Characteristics of the product	Scale; Direction; Coverage (Ledeen 2011); Commoditization, flexibility and change; Scale and complexity (Daneshgar et al. 2011)
Characteristics of the organization	In-house information systems expertise; Operational factors; Support structure (Daneshgar et al. 2011)
Situational factors	Total Cost of Ownership; Timing; Standards (Ledeen 2011); Cost; Time; Risk (Daneshgar et al. 2011)

Although the discussions had in the literature have similar themes, the notions in Table 2 are given without any case examples. The literature does note that not all the criteria are relevant in every context. Daneshgar et al. (2011) observed, for example, that small and medium enterprises are less likely to make the decision based on the strategic aspects or for acquiring intellectual properties. Additionally there might be issues regarding the context of the choosing process in the literature; usually build or buy literature refers to large scale acquisition projects for example with enterprise resource planning (ERP) systems. If smaller products are not taken into consideration in these articles, there may be problems in applying all the criteria to cases where the scale of the products is not as massive as for the ones they were created for. There were, on the other hand, criteria regarding scale and complexity of the product, and therefore the literature sources may be considered to discuss in rather a general level. Finally, because the general nature of these points, these criteria can be used in testing tools context, but some would argue that in most cases building own tool is not a great choice (e.g. Craig and Jaskiel 2002, p. 227).

The reviewed literature regarding the decision whether to build or buy often highlights the strategic significance of the tool for the organization. In his discussion regarding the criteria whether to build or buy software systems, Ledeen (2011) recognizes first the need to identify whether the role of the product is core or context for the company. If the product is significant from strategic point of view, i.e. is core, for example being very close to the own core competence and "contribute directly to the organizations differentiation and value creation", developing the product itself may be beneficial. Similarly Langer (2012, pp. 42-43) discusses that the strategic weight of the product is

very critical in decision process, referring to drivers and supporters. A driver functions would be activities that are essential for the department to contribute to the goal of the whole organization and applications for it should be developed in-house, while supporters are activities that do not generate directly revenues, such as accounting, and necessarily do not need to be built in-house. In their literature review regarding larger organizations, Daneshagar et al. (2011) also recognize that strategy and competitive advantage is one factor when deciding on between build and buy, suggesting that less strategic applications should be purchased whilst more strategic developed internally.

In testing tool context, Craig and Jaskiel (2002, p. 227) recognize that the only reasonable situations in which the product should be developed internally are when the organization is unique or when the organization wants to capitalize on existing expertise or infrastructure. If the area of business where the tool is implemented is not core, it may be even reasonable to consider modifying the organization's practices to meet the restrictions of a software product (Ledeen 2011). Craig and Jaskiel (2002, p. 227) illustrate the core and context discussion saying that developing own word processors, for example, for the organization's needs is not reasonable. However, it is noted that if the organization wants to own the intellectual property of the product, they may need to develop it themselves (Daneshgar et al. 2011).

The literature suggests evaluating different aspects or characteristics of the product when considering whether to build or buy. Daneshgar et al. (2011) note that the complexity of the product relates to the decision regarding whether to build or buy; if the product is straightforward, it can be easily developed while if the product needs specialized technologies, buying the solution could be more beneficial. In coverage section of his discussion of whether build or buy, Ledeen (2011) notes that not all products fulfill every requirement, and while the product may be able to fulfill most of the requirements, it may have additional features that the organization will not need. This is seen unwanted, as it gives the product higher costs and more complexity. This in turn may require more resources to maintain and operate, leading to a situation in which fully customized internally developed product may be more beneficial (Daneshgar et al. 2011). Ledeen (2011) also recognizes that while the considered package may fulfill the current requirements, it needs to be flexible enough to fulfill the future requirements, as well. Mature and standardized products may be bought and products with unique requirements should be developed internally (Daneshgar et al. 2011). The scale of the desired product may be considered a factor; the larger the application more risks and costs it produces (Daneshgar et al. 2011; Ledeen 2011). Craig and Jaskiel (2002, p. 227) add to this discussion the fact that an in-house developed solution needs to be tested, maintained, and documented, which may require lots of resources. It may be argued that more complex and bigger software products have greater need for these activities.

Daneshgar et al. (2011) discuss that the organization for which the product is implemented affects the build or buy decision. Firstly, the organization's background on making or buying the tools used may have influence on the decision making (Daneshgar et al. 2011). Hence, culture of organization may even influence towards wrong decision because of the lack of knowledge in the other. Secondly, in-house expertise on building information systems may become a great factor when deciding whether to buy or build. If the organization has enough expertise on building the desired package, it might develop it cheaper and with more benefits. On the other hand, it also must be considered that the skilled developers are not available to attend in other development projects during that time. (Daneshgar et al. 2011) Finally, the support structure, that is what existing technology the organization has already in use, can be considered a factor. If the organization is renewing systems, the new system needs to be well integrated with the set of software products that was used with the old one. A bought package needs in this case heavy customization, and internally built solution may be customized specifically to the environment. (Daneshgar et al. 2011)

Final factors recognized in the reviewed literature influencing decision making are what in this study are called situational factors. Cost can be listed as one of them, as the impact of cost may differ depending on the situation. Ledeen (2011), for instance, would list total cost of ownership as one inspected criteria for buy versus build, noting that there are multiple types of costs, such as acquisition, configuration, customization, and maintenance. Also Craig and Jaskiel (2002, p. 227) recognize that the cost is not only the licensing costs, but also other hidden costs in implementing and training, which may very well exceed the licensing costs. Daneshgar et al. (2011) likewise note that implementation and ongoing costs are factors influencing whether to build or buy, even though it has become less important for organizations. Another situational factor is timing. Although one could think that implementing commercial package could be faster than custom development (Daneshgar et al. 2011), it is not always true (Ledeen 2011). If the organization needs to adapt to the system or the developed product needs to be customized, it takes longer time. The less modification needs to be made to adapt to ready solution, the more it has time advantage to the internally developed tool. (Ledeen 2011) The whole lifecycle of the product should also be considered when making the decision. If the system requires commitment for a longer time and provides lasting value, it may be beneficial to develop it (Daneshgar et al. 2011).

3.5.2 Evaluating software systems

Ways to evaluate commercial software product from the buyer's perspective is discussed in some literature sources. Often in the literature the bought software products are called commercial-off-the-shelves (COTS) products (e.g. Carney and Wallnau 1998). Correctly evaluating software systems may be regarded important task when choosing between different tools, as choosing wrong system after decision process may result in economic losses to the organization (Jadhav and Sonar 2009). It has been noted

that evaluating commercial software system is not an easy task, and it needs considerable effort (Majchrzak 2012, p. 47). Carney and Wallnau (1998) note that even "—with perfect knowledge of COTS products and system requirements, COTS evaluation would be difficult", while emphasizing that being a level of knowledge that never can be achieved. They describe the evaluation process having trade-off decisions among criteria, some of which interact with each other. A design of a product may be fit in one way, such as in functionality, and unfit in some other manner, like in performance. Coping with these limitations may be regarded a part of the evaluation.

The literature identifies basic or commonly repeated structure for acquisition of software product. In maintenance tool acquisition project, the three main steps according to Khan et al. (1997) are definition of goals, gathering information about the tools, and selection of candidate tools for evaluation purposes. Carney and Wallnau (1998) rather similarly recognize that prior selection of software product there are three steps that make the decision-making possible: identifying alternative courses of actions, defining criteria to assign a measure of merit, and assigning that measure of merit to the selected alternatives. These steps are followed by the actual decision regarding which of the software products is chosen. The literature review by Jadhav and Sonar (2009) shows steps that take more into consideration the selection of evaluated products. These steps include determining the need for purchasing the system and investigation of the availability of the required package, listing of candidates, eliminating candidates based on lack of a feature or restriction that is unacceptable, using an evaluation technique on those candidates, doing further research on tools by conduction an empirical evaluation, negotiating a contract, and finally purchasing the most appropriate software. Similar steps are noted in more practical context, as well. A white paper from Traq Software Ltd. lists five phases for selecting a test management tool. It starts with identifying the problem, as in what is currently wrong and what needs to be fixed, and is followed by assessing the appetite for change, in which the desire for change is validated by other team members. The process proceeds to listing potential tools, in which requirements are compared against the offerings of the tools and ruling out clearly unsuitable alternatives, and selecting the front runners, in which the requirements are evaluated with a grade and a weight depending how relevant the requirement is. Finally the process ends with a trial phase, in which the tool is used in order does it work in practice. (Traq Software Ltd. 2011) As a summary, the literature seems to suggest first to identify the needs, requirements, and criteria for the tool evaluation, listing different alternatives, evaluating the alternatives in one or two phases against the criteria with increasing precision, and finally choosing the most suitable tool. These steps are validated in other literature sources as well (Poston and Sexton 1992).

The literature provides systematic methodologies or evaluation techniques to help with the choosing process, usually including some sort of prioritizing of attributes or criteria. The Multi-Element Component Comparison and Analysis (MECCA) method uses hier-

archical attribute relationships to calculate and understand the meaning of different attributes (Khan et al. 1997). For instance, if one highest level attribute for testing tool is usability and it is weighted with 40% of all high level attributes, one should examine of which lower level attributes the usability is consisted. These might be test execution usability and test scheduling usability with weights 60% and 40%. By giving values to all the lowest level attributes with numbers from 0 to 10, one can calculate the final score for the tool. Rather similarly a white paper from Traq Software Ltd. (2011) provides technique involving weighing attributes to evaluate how well test management tools fulfill different criteria. In their model, the rating is done by weighing the tool feature, e.g. usability or being web based, rating with a relevancy coefficient ranging from 1 that is essential to 0 to being not required. After different feature ratings are given to all features, they are summed, and the tool gets the total score, which can be compared to other tools.

The literature identifies attributes or criteria a major element in evaluation process. Furthermore, the literature suggests some groups for criteria with which the software products can be evaluated. In their literature review regarding general software product acquisition, Jadhav and Sonar (2009) note that there are certain groups of criteria suggested in the existing literature. The identified criteria groups most used in literature are the functional characteristics and the quality related characteristics. The latter included many criteria such as personalizability, portability, maintainability, usability, reliability, and efficiency. These criteria groups may be regarded to follow the basic classification of requirements in software development: functional and non-functional or quality requirements, such as usability, efficiency, performance, space, reliability, and portability (Agarwal et al. 2010). For other criteria for evaluation, Jadhav and Sonar (2009) identify vendor criteria as well as cost and benefits criteria. They also list less common criteria, such as hardware and software, opinion and output related criteria.

In testing tool selection context, reviewed literature suggests similar criteria and considerations for selection process like in more general context. Illes et al. (2005) list nine quality criteria for a software product. The main groups listed are: functionality, reliability, usability, efficiency, maintainability, portability, general vendor qualifications, vendor support, licensing and pricing. Poston and Sexton (1992) note that there are general criteria, such as productivity or quality gain, environment-dependent criteria, such as platform changes and organizational changes, tool-dependent functional criteria, as in single features the tool must have, and non-functional criteria, such as performance and user friendliness. Majchrzak (2012, p. 47) note that in addition to price and performance, organizations should take into account integration capabilities, effects on the processes as well as other impacts on organization. Parveen et al. (2007) recommend that when choosing a test management tool, one should keep the future in mind. With this, they probably suggest to forecast possible changes and seeing if the tools may be modified to match these changes. Craig and Jaskiel (2002, p. 221) note that the tool

should be modified to the needs of an organization. They also recognize that “another major issue in tool selection is ease of use and technical bent of the testing staff.” Finally, they discuss the importance of vendor and recognize that the responsiveness of the vendor is a key factor when selecting testing tools while having multiple tools from the same vendor may be beneficial.

Some of the authors discuss the importance of specifying environment in which the tool is implemented when choosing testing tools. Craig and Jaskiel (2002, p. 219), for one, state that for “-- the most part, it's not a good strategy to choose a tool and then modify your procedures to match the tool”, and usually it is necessary to first define the process and after that choose a tool for that process. Furthermore, it is noted that it is not possible to evaluate software product without the context or environment in which it will be implemented (Carney and Wallnau 1998). Similarly Poston and Sexton (1992) note that testing may be regarded differently in different organizations and the evaluators need to identify what testing activities are involved in the context of the company. Not knowing or acknowledging the environment and functional criteria may produce problems in tool selection process. Craig and Jaskiel (2002, pp. 219-237), for instance, note that if the organization has no clear strategy or idea what the tool is for, it may end up choosing the wrong tool.

The literature highlights the importance of functional criteria for the evaluated testing tool. Parveen et al. (2007) note that there should be enough time for all the functionalities to get thoroughly covered, as limitations of tools may discourage the employees to adopt them. In their case study, the limitations of selected tool made it unusable to some team members. Also Khan et al. (1997) recognize that the main functionalities of these maintenance tools should be examined in order to evaluate them. Some of the functionalities may revolve around software retesting functionalities, knowledge redocumenting, information repository, display function, and tool interconnections. To get all functional criteria for their testing tool, Illes et al. (2005) analyzed the test process. The main reason for this was to point out different tasks among different roles in testing process and to use this information to formulate functional criteria for a test management tool.

It can be seen from the literature that no matter the context of the organization, it needs to weigh on functional and non-functional criteria when evaluating different alternatives. Unsurprisingly the functional considerations are visible in every literature source, as it is the functional requirements that mostly fulfill the needs of the organization. Non-functional criteria include various characteristics revolving around usability, customization, performance, the vendor, costs, and portability. In testing tool context there are similar themes in non-functional requirements, but integration and customizability needs are specially highlighted as areas of interest. Possibly testing processes may require the increased interoperability with other tools and adoption into the organization’s processes. These aspects may be valuable when using the tool as they terminate how the

tools work in the organization, and without necessary attention, the tool may be a bad fit for a testing team.

4 RESULTS

The results chapter of this study will introduce the results of the empirical data gathered for the research. Firstly, the procedures and activities to manage testing in case organization are briefly introduced in order to get an overview how test management is done in the organization. This is followed by exploring what difficulties the organization has in doing its test management activities. This is done by using interview and observation data acquired during the project and using the terminology used in the reviewed literature. The third part of the results will introduce the criteria developed in the case project for choosing a test management tool. Finally, the evaluation of different alternatives is done against these criteria. This chapter uses mostly the interviews done and observations had in this study as its source material, and they are notated similarly as the literature sources (O6) and (I1). These can be found in Appendices B and C.

4.1 The case organization and testing

As the author has worked in the testing process himself, the initial process description was done from memory and through observation. Additionally, several internal documents were used, such as test plans and checklists. This approach had certain limitations, such as relying only on one's memory and seeing only from the author's perspective. In order to reduce these limitations, the descriptions were validated and confirmed by other employees involved in the testing process.

The case organization has established practices that are followed when the software is tested, but they are not documented thoroughly or the documented procedures are scattered in different documents. Test management in the case organization was mapped in order to understand what test management tasks are done in the company. In this study, there are two main purposes to establish the process and test management activities. Firstly, exploring testing activities helps to understand the environment for which the tool will be implemented; it is a key for communication regarding what the tool is trying to enhance and what kind of restrictions the process has. Secondly, the mapping will provide information regarding the possible problems in testing and the difficulties the tool is trying to solve. In the literature review, test management was defined as being a discipline involving test process, testing assets, and testing team. Test management in the case organization is researched from these perspectives. However, as the ultimate purpose of this section is to provide information about the tool criteria, the inspection of test management in the section is limited to the organizational and technical aspects, leaving other aspects, such as leadership, out of it.

4.1.1 Overview of the case organization

As the aim of this study is to increase understanding on the subjects of the study especially in the context of the case organization, it is important to understand in what kind of environment the organization operates. M-Files Corporation is a Finland-based software company developing enterprise document management systems mostly for corporate use. The company has offices in Tampere and Espoo in Finland as well as Texas in the United States. It also offers various consulting services to help the usage of their product and other customer software development projects. Big actors in document management system product sector and competitors for M-Files are, for instance, Microsoft, IBM, and OpenText, that can be considered market leaders (Gilbert et al. 2012). At the time of this study, M-Files has got over 130 employees, 27 of whom are in research and development, placing it among small and medium enterprises. The customers are from a wide range of industries, such as pharmaceutical, manufacturing, real estate, and accounting and finance. For this broad customer segment, the company requires certain quality in its product, as it itself sells a product variant called Quality Management System. (M-Files 2013) There has been increasing need for visibility in the quality assurance processes, and it has been seen that customers may start to need more information about how the quality of product is assured (O13). The corporation strives for better quality with standards, such as ISO 9000, which it acquired during the time of this study. The aim of the company is to grow radically and become more internationalized. For few years it has worked, and the revenue of the company was approximately 8 700 000 euros in 2012, which was over 44% more than year before. (O1)

The main product of the case organization is M-Files, which offers file system-like behavior fully integrated to Windows environment with extensive customizability by providing possibilities to create a structure and relationships between different elements, such as documents or customers, according to the business process and the real-life environment. The system may be further customized with various application programming interfaces (API) and scripts. The customization can be done with consulting services provided by the organization or by the customers that have sufficient programming expertise.

The software product M-Files is updated every year with a new version, in which the old features are enhanced and new features are introduced to a new release. The development cycle usually starts and ends around July or August. This is very different from customer project-based software development, in which projects may be very different from each other. The type of software development process, which includes developing a product and afterwards updating new versions of it, is called software maintenance (Agarwal et al. 2010, pp. 35-36). From testing perspective, this means that in addition to testing new features, the organization needs to test the old features in order to assure that they work after a new functionality has been implemented. Some of the features

may be many years old. Most of the research and development as well as testing is done in Tampere office, which makes coordination simpler than in distributed development.

4.1.2 Testing process

Like the literature review suggested, it is beneficial to map the testing process in order to understand which aspects the tool should facilitate. Not unlike the search for test management tool requirements in research by Illes et al. (2005), the testing process is mapped to find out some of the functional criteria that could help the organization make its decision which of the criteria are the most important. Unlike the research by them, this study aims also to find out and understand the problems of the current areas related to test management.

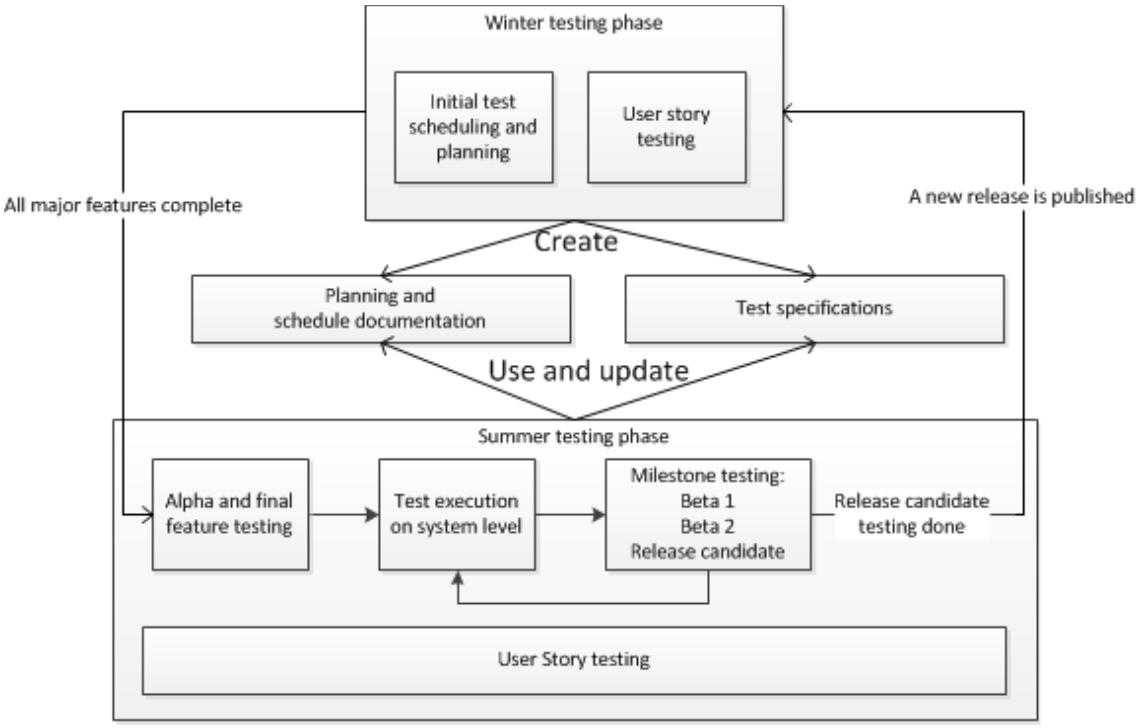


Figure 9: Overview of testing process in the case organization. Only testing related phases are included in the testing process.

Testing of product M-Files is roughly divided into two distinguishable phases. The "winter testing phase" includes User story testing and preparing for summer testing, and it starts around the time of the release of a new version of M-Files and ends roughly when all major features have been implemented. The other phase, "summer testing phase", includes final feature and large-scale regression testing during the summer from final feature testing to the release of the product. The phases are visualized in Figure 9. The aim of the winter testing phase is to incrementally test the new features made to the product, while in the summer all the new features are retested with some of the old features. Both the phases have User story testing, but during the summer there are less of it. (O2) For this study, the process is described more specifically, starting from implement-

ing new functionalities and features into product and finishing with a tested new version of the product.

As the organization develops software using agile methodology Scrum, testing is done for each new increment separately. One increment is called a User story and it is usually implemented during one sprint, which lasts two weeks. Comprehensive visualization of User story testing process is included in Appendix D. This swim lane chart shows both the different activities in testing process done by different roles and the most central artifacts used. First, the scrum team selects the User story to be implemented in the sprint and who does the implementation. After that, the developer assigned to the User story will ask for a tester from test manager by e-mailing some information about the contents of the story to testing pool mailing list. Test manager in turn checks which tester is able to take on the User Story and informs the tester. If no testers are available, the test manager will put it on backlog, summary of which is reported in the testing pool mailing list regularly. When the tester has been assigned to a User story, software tester and developer will work on the User Story from that point on.

Testing phase for the User story starts almost immediately after sprint planning. Optimally the tester and developer start to plan how the testing should be done and what scope should be covered after the developer is assigned with a new User story. This phase is called test planning, in which the tester will generate general test cases or at least themes that will be included in the actual testing. This starts with a discussion between the developer and tester. Although the test planning discussion should follow sprint planning meeting, often the developer needs to further specify what tasks the User story will include, and test planning will be done a bit after that. This might be the case as well when there are no free testers in the team.

While the developer implements the functionality specified in the User Story, the tester writes test cases into a test specification document that will be expanded and executed when the functionality has been completely or nearly completely implemented and included in M-Files build. Usually the test planning phase takes less time for the tester than the developer, so he or she might have other user Stories or testing related activities at the same time. After the new functionality is implemented and a working build, i.e. a revision of the product, is delivered to the tester, he or she will execute the created test cases manually in order to verify that the User story is implemented with sufficient quality. Usually the tester will also update, add, and complete test cases in test specification document to match the realized functionality. At the same time, the tester executes these test cases by exploring the implemented feature and records or updates the status of test case in the document (O3). The steps describing how some of the test cases are executed are also included in the test case at this time. All the problems or misfunctionalities found during testing will be discussed with the assigned developer or added straight to the defect tracker as a new issue, in which the User story ID is referenced.

The developer either fixes the defect found during the testing or discards it with reasonable explanation. (O3) All the fixed defects should be retested (M-Files 2012, p. 15). Additionally the developer and tester should go through the created test cases in order to verify that all the necessary scenarios have been covered (O3). When the tester decides that the User story is finished, it can be regarded as tested from the tester's point of view. From the developer's point of view, the testing of User story will be done after all the possible automatic unit tests, i.e. NUnit, and automated regression tests have been executed. After both manual and automated tests are executed, the User story may be regarded as tested. A new User story will be started until all the new features have been implemented.

After all the main features are included in a build of the product, begins the system testing phase that is be very important for the overall quality of the released version. This is usually called summertime testing, and it includes activities done to ensure that the new features work as whole and that no regressions were introduced while implementing the features. System testing phase may be regarded as started when Alpha testing begins, in which all the new functionalities are tested with the existing and possibly new test cases. Basically the system testing includes Alpha testing and system testing as well as milestones Beta 1, Beta 2 and Release Candidate, each of which has a day or two of verification testing. In each one of them, the test manager specifies certain sets of test cases to be executed manually. These test cases are located in a test specification document.

The tasks in test specification testing are visualized in Appendix E. The test specification testing regards system testing activities, in which the existing test cases are executed. In this process, new test cases may be added and deleted as well, but usually no new test specifications are created, as the emphasis is on test execution, not improving testing assets. First the tester will find the test specification assigned for him or her from testing schedule. With the newest or in some cases a specified build, the tester will execute manually the steps in test cases and compare the actual behavior against the recorded expected result. If other than the recorded functionality is recognized, the tester will check for existing issues that would match the found defect from the defect management system. If a similar defect is found, the tester will update the existing issue with the build information and possibly other relevant information. If no similar defects are found, the tester will create a new one and add its ID to the test specification. When all the test cases have been executed, the test specification will be added with testing duration. After this, the tester will update the testing schedule with a marking that the specification has been tested and continues with the next test specification. When the last test specification has been executed, the phase is considered to have ended.

The current strong points of testing process were discussed in the interviews briefly. It was noted that currently it is easily for testers and developers to communicate with each

other (I1; I2). This makes resolving the actual functionality of feature easier. System-wise it was noted that defect tracker system was responsive and easy to use (I2). Senior software developer noted that the advantages of the current processes include test execution being efficient, the test case execution is becoming more automated, and the preparation of testing is started early on before developer has developed any functionality (I4). Having agile development in organization was seen an advantage from testing point of view. Test manager noted that testing the new functionality by User story elements is a good thing (I3). For this study, these strong points are kept in mind in order to overcome challenges in the testing practices without losing its advantages. This means that a change in testing process itself is not needed, defect tracker is sufficient, and that communication can be used to complement some limitations in test management systems.

4.1.3 Test management elements and activities

Test management was not part of the vocabulary in the case organization before the project. However, there have been many activities that can be regarded as part of test management. Test manager had been planning and coordinating the testing but said that test management has been "invisible" work (O13). Because the aim of this study is to improve current practices but not change processes significantly, the elements and activities regarding test management are introduced in this section.

In M-Files, test management activities may be regarded to include planning, prioritizing and specifying the final features and regression testing activities for the summer testing, prioritizing User story testing activities during the winter, leading the team, monitoring and reporting the testing activities, and rescheduling plans throughout the year. The reporting of results will be done with the automatic reports in M-Files vault, and manually through different means of communication. These plans are mostly done by discussing with developers and subjective evaluation of risks. The approximation of the needed time for new feature testing is based on previous experiences and comparisons to previous features, discussions, and intuition. Usually all the new test cases are rerun during the summer. (O13) Testing is executed until the release date, and as many test specifications as possible is executed before that (O6). Release date may be changed depending on the observed quality of the product. The quality is derived from the amount of open defects and overall feel of how the version works (O13). This means that the time and the number of defects are the main criteria when to stop the testing. The schedule is used to monitor which test specifications have been executed (O13). Currently there does not seem to be a way to see which test specifications actually were executed to find the defects, though (O6). So, in fact, the number and severity of defects are part of the release criteria, but there is currently no easy way to see how many tests were executed in what kind of environments.

In the literature review it was recognized that test asset management is a great part in test management. Like many other documents in the case organization, most test artifacts are housed in the M-Files document management system. A major benefit of using it is that the product allows the documents to have history data, so that after the changes are made to a document, the old versions are found in the program if needed.

Test management regards the means of creating and housing different testing artifacts. These artifacts include the documentation regarding the plans and schedules of testing, test specification documentation, and test reporting. In the case organization, one of these documents is the test plan, which includes the overview of what is tested, how everything is tested, and what is needed for testing (O5). Another central artifact is test schedule, which is an Excel spreadsheet including the testers and test specifications assigned for each day (O4). The test schedule is planned by test manager and the status of it is updated by testers. The found defects are recorded as Issue objects in M-Files vault called Tracker, which is designed mainly for being defect management system. All these artifacts are housed in M-Files document management system so that everyone can find them easily.

Through observation and the process mapping, it can be seen that a central testing artifact in case organization is test specification. A test specification is a Word document, in which there are numerous test cases. A new version of the test specification is recorded in the current system usually when the tester has finished testing the set of test cases in that specification or when test cases are added, removed or modified. The document also includes some information about test environment, usually the build for which the cases were executed previously as well as the language and the operation system. These test specifications include all the information of test cases and how the latest test execution ended. Test specification also includes a marking of the IDs of defects in a separate defect tracking system. The overall design of a test specification is forced to be quite similar, but there are many documents that differ from the template, as those are changed over time.

Measurements of testing process are gathered by using the metadata of test specifications and defect objects in Tracker. The main metrics gathered are number of test cases created, run, and failed, number of test specifications, number of defects found in each feature, and time used for manual testing (O5). The number of test cases created, run, and failed in testing as well as time used for manual testing are added manually to test specification metadata by estimating the time spent and counting the test cases in the document, while number of test specifications and number of defects found in each feature is more automatically gathered by counting the number of objects in M-Files. (O6)

In a way, the system in case organization for maintaining testing documents and history data of them can be seen as a test management tool. It provides some means to house

test cases, run data, and test plans. The different tools in this system are Microsoft Word, Microsoft Excel, and M-Files. From investigating the current test management system in-depth, it was noted that it has similarities to the literature. As Majchrzak (2010a) noted, a test management tool may be in some level, from the lowest of pen and paper to dedicated automatic software systems. In the case organization, the means of managing testing effort and artifacts seem to have evolved beyond the ad-hoc pen and paper or communication through e-mail to the level of using common Office tools such as Word and Excel and M-Files to distribute to team members and add other information to documents (O6). As noted in literature review part regarding test management tools, this is one way to enable test management in an organization. However, having more dedicated management tool might enhance the organization's capabilities to plan, monitor, and report testing based on accurate information without the need to do manually-intensive tasks. Therefore improving current test management systems is justified.

During the case project, it was noted that test cases and defect tracker issues had a great value in terms of documenting the functionality of the product. It was observed and discussed with test manager that these elements are in fact the only documentation to clarify how the software product works in detail level (O6). Hence, it may be argued that having test cases up-to-date and visible for all stakeholders is important. One software developer had expressed that he sometimes checks the functionality from test specifications (O6). Having information available for everyone may be seen beneficial if someone needs to validate that the product is working correctly. However, it was noted that currently it is difficult to find test cases, as they are inside Microsoft Word files (O6).

4.2 Organization's challenges in test management

Using the reviewed literature, the challenges can be grouped under test management activities and challenges recognized in them. It can be noted that the case organization has various problems in test asset management and metrics gathering, and mostly because of those, in test management activities relating planning, monitoring, reporting results. Additionally a central reason for various problems is the system having test specification as an element for planning and execution. The central problems are visualized in Figure 10. According to several interviewees, the current testing process itself is not problematic (I1; I5), and test execution is seen rather fluent and effective (I4). Therefore difficulties in these interviews often focused rather on the difficulties on test asset management and execution monitoring, than problems in testing practices. As test process was not seen problematic, one interviewee stated that the next step was to enhance the tool support (I5).

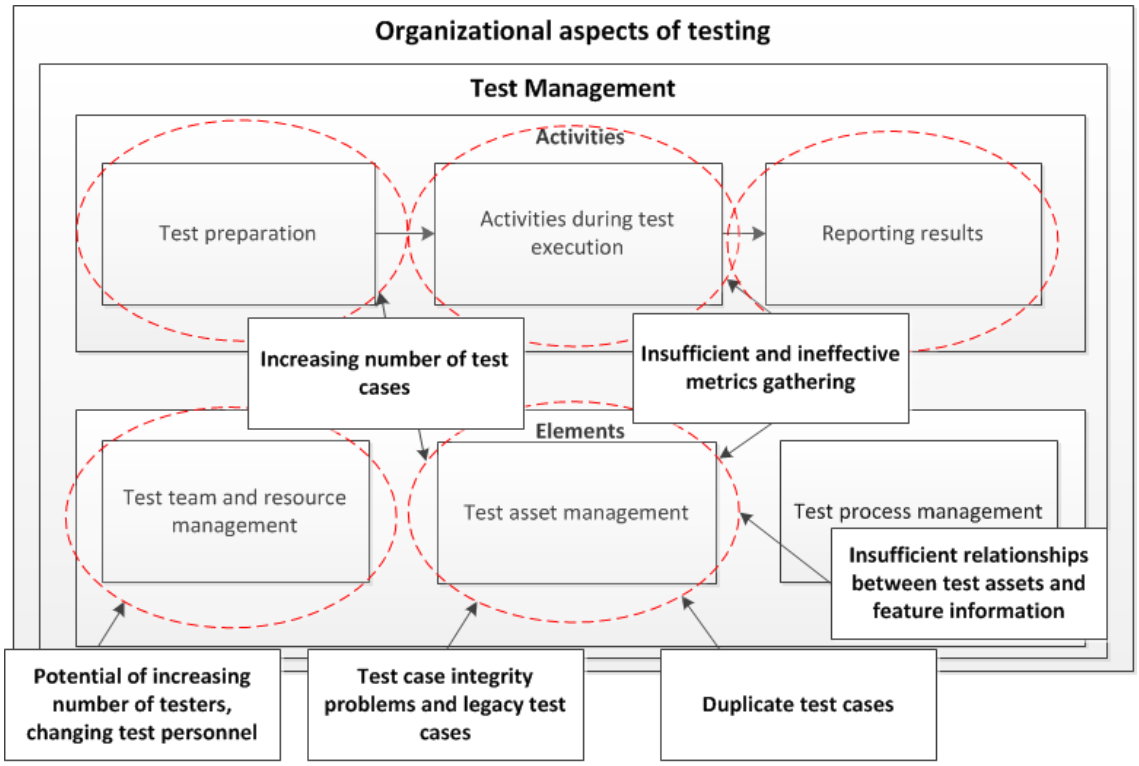


Figure 10: The key challenges in managing testing in the case organization in relation to different aspects of test management.

Some topics were clearly more frequently discussed and observed than others. The challenges are listed and compared in Table 3. The table shows that the team saw that there are various challenges related to test asset management. In fact, six of the eleven listed challenges are linked to test assets somehow. Challenges in planning and monitoring activities were apparent in both interviews and observations, but reporting results was not discussed in any of the interviews. Thus, it can be said that while there were difficulties in most of the aspects in test management, challenges in test asset management were clearly the most dominant.

The combinations of the interviews and the observations connected to a specific challenge show who has noticed the problem. However, the Table 3 does not depict the severity of the separate challenges very accurately, as it shows only how many people have noticed the problem; a challenge may have been mentioned only in one interview because it really is visible only to one role in the team, but it still may have great an impact on the testing process. On the other hand, the observations may be influenced by the literature, and the rest of the team may not have noticed the issue, which could lessen the severity of the problem. Thus having a team member to identify the problem can be seen to increase the severity of the problem a bit.

Table 3: The challenges and their sources in the case project.

Test management aspect	Challenge	Sources
Test asset management	Management of test case information in the current systems is difficult	I1; I2
Test asset management	Duplicate test cases cause the process to be less effective	I1; I3; I4; O6; O13
Test asset management	Legacy and other outdated test cases confuse and cause additional work	I1; I2; O6
Test asset management	Test case integrity confuse what to test	I1; I2; O6
Test asset management	The systems do not show the relationships between test assets	I2; I3; I4; O6
Test asset management	The amount of test assets increases	O6
Test team and resource management	Number of testers may increase and the personnel may change	O6
Test preparation	Plans and schedules change often	I3; I4; O6; O13
Activities during test execution	Metrics gathering is difficult or impossible	I4; O6
Activities during test execution	Monitoring current status of testing is difficult or impossible in low detail	I3; O6; O13
Reporting results	Results cannot be reported with data	O6

Problems in current systems to house and manage test case information became a central theme when discussing difficulties in current test execution in the interviews. Especially testers noted that the current test specification system is either difficult (I1) or outward horrible (I2). Testers were especially worried about the management of test cases. There were various notions about problems regarding duplicates (I1; I4): during User story testing similar or the same test cases are sometimes written in different test specifications, and if the functionality changed, test cases needed to be updated in every place those were created. There is a possibility that some places were forgotten, and the old version had remained in the test specifications (I1). For many times when developing new test cases for User stories or during the summertime testing, there is an apparent risk of creating test cases that already exist in some other test specification. A few times during the project there was a need to remove duplicate test cases from some test specifications, demonstrating that these duplicates are troublesome and require extra effort from the team. (O6)

During the time of this study, it was noted that there is a great amount of duplicates in different test specifications, but the number of them is not known (O13). In addition to being hard to update, the duplicates make it hard to see when a test case or a duplicate

of it has been run last (I3). Duplicate test cases are troublesome in managing testing, and they are seen undesired in the reviewed literature (e.g. Neto et al. 2012; Iberle 1999). Finding and removing duplicates could add more efficiency and remove costs from regression testing, like noted in the literature review. However, with the current systems, it is not feasible, as every test case needs to be checked against every other test case in the system. It also would be reasonable to assume that duplicate or very similar test cases hinder the team's ability to control the testing effort; when only some of the test cases can be run during regression testing, one needs to make sure that no additional time is spent on test cases that have already been executed.

Test case integrity or the completeness of test cases seemed to be a problem in the case organization. Firstly, legacy test cases, i.e. test cases created during testing previous releases, stored in the systems appeared to be problematic. According to one tester, it was often hard to understand what parts of test specifications were still relevant (I2). Additionally if someone wanted to see information regarding a specific test case, he or she would need to look history information of the whole test specification, which is very time-consuming (I1). These legacy test cases seemed also to be written often in a less than desired manner without exact steps to execute it or expected outcomes (O6). This makes knowing the real functionality and the exact way of getting the functionality shown difficult. Finding the employee who created the test case may also be difficult, as one needs to view all the old versions of test specifications to find the information (O6). One tester noted that he does look information, such as who has executed the cases earlier and what with outcomes, in the version history, but doing it is not easy (I1). Problematic with the version history is that the changes in test cases and executions are in the same version, i.e. all the test cases may have been executed in one version, or only some of the test cases were changed but not run, or a combination of these two (O6). In other words, if one needs to find a change in test case or results of test case execution, he or she needs to look through multiple versions as the changes in test cases and test executions are not separated. Relatively new test cases may have similar problems in changing functionality, as well. This was witnessed during the summer testing, when the developer was on vacation, and the functionality was not like discussed during User story testing. Afterwards it was revealed that the functionality behind the feature was changed, and the test cases were not updated. (O6) This incidence shows the problems in having two separate and changing elements, test cases and features: if one changes, there is no guarantee that the other one will be changed, as well. This usually leaves the test case with old and incorrect information.

When looking at different test specification documents made by different employees at different times, it becomes apparent that the documents are not consistently written. Some of the cases missed steps, expected outcomes, and in some cases the test case may only have a name stating what should be tested excluding any information how to do that. This was common in both newly written and legacy test cases. (O6) The current

status of test cases in the organization was discussed briefly in many interviews, as well. Test cases were noted to be very fundamental element of testing in the organization (I5), but the current implementation of them is not without problems. Testers noted that steps how to execute test case are not explicitly recorded in every case (I2), and expected outcomes of old test cases are sometimes based on the memories of testers and developers (I1). Test case system cannot be therefore regarded as complete by any means. This was noticed during the summer testing phase, when new testers had troubles with steps or expected outcomes. This took a great amount of time from the other testers and developers. (O6) Current test specifications rely a lot on the software tester's understanding of the system, which becomes increasingly more complex in each version. If the personnel suddenly change, it can be hard to utilize these existing test cases, as the person who made the case cannot be contacted. Furthermore, not having well documented test cases limit the organization's possibilities to outsource the testing effort, because the other organization would need a lot of knowledge of the product in order to test it.

Additionally to the management of test cases, the interviewees discussed the current methods of having relationships between test assets being insufficient, causing various problems. One of the testers noted that with the current system there is no way to see from which features the defects are found, and if one wants to test the functionality from which numerous bugs are found, there is no way to find these functionalities (I2). Test manager also noted this problem, stating that this information gained by having relationships between test assets could be used to plan regression testing (I3). The relationships between elements like requirements or features were noted also by the senior developer, who recognized that the inability to measure coverage, i.e. seeing if there are enough test cases for each feature, is a major problem with the current system (I4). Without these relations, it may be hard to see the whole picture of testing effort. It was noted that one can see that certain test specifications are executed during the summertime testing on the schedule, but there may not be a way to see how many test cases are run to assure that a certain feature is working as intended (O6). The number of test cases run on one feature may not give a full understanding is it working properly, though, but comparing it to total number of test cases on that feature may hint is it even nearly tested. This could help planning and organizing testing effort even more.

The amount and complexity of test data might increase and thus become more problematic in the near future. As the testing effort grows with the complexity of developed product, it would be reasonable to assume that the testing team grows and the communication regarding the testing effort and artifacts may become more difficult. The developed software is the main product for the company, and hence it is developed aggressively for many years to come, probably without removing any features from it. The testing effort will hence increase in every version. The team dedicated only to testing had already grown from 4 in 2012 to 7 in 2013, and new testers were going to be hired next year (O6). The need for communication could be reduced by introducing guide-

lines and formal processes. Alternatively if the organization does not increase the number of testers, it has to cut down the amount of testing somehow. One way to do this is to further automate test case execution. Another way to do this is to selectively choose the tested areas by using various regression testing techniques introduced briefly in this work. This, on the other hand, would need detailed information about the changes and increased traceability, which the current infrastructure does not provide.

In addition to the increased amount of testing data, increased need for communication may become troublesome when the testing and development teams grow. During the observed period, there were no issues with the e-mail system, but there were multiple occasions in which test manager had concerns regarding the increased amount of e-mails requesting testing or test planning. The test manager needed to have an ad-hoc backlog in a text file and remind herself of the upcoming tasks. (O6) Although at this point this did not seem very laborious, the growth of development and number of User stories will likely make monitoring and controlling User story testing this manner very laborious. Possible new testers may not have similar experience of the system under test as the old testers, and as the test cases may be vague, actual manual test execution may take longer time than it should.

From test management activities point of view, the interviewees recognized difficulties in planning and scheduling test execution. In the planning phase, test manager noted that estimating testing effort is hard even with the old features and test specifications. Previous time estimates may not be accurate as it depends on the tester. One tester may have included all the meetings and other activities that are necessary but need time, and another may not. (I4) Also, if many testers have been executing one test specification, estimations can be incorrect (O6). It is also noted that planning, scheduling and re-scheduling activities take a considerable amount of time with current Excel based tool (I3; I4). Using Excel spreadsheet needs a large amount of caution from the planners, as well. Because Excel spreadsheets and test specifications are located in different systems, there is a chance that when the test specifications are shifted from one cell to another some of the test specifications are lost from the schedule (I4). Also, it was observed that even during the summer more test specifications were created and needed to be added to the schedule (O6). If for some reason a test specification is missing from the schedule, it may not be run. At one point this problem was manifested as a test specification was divided into multiple ones, and the divided specifications were forgotten to be included in the schedule (O6). Test manager stated that someone needs to check that all the test specifications are in the schedule during the summer. (O13) This adds manual activities that take time from actual test execution.

Like mentioned in the literature review, many test management activities were facilitated or even enabled by the use of metrics. Through observation, it can be seen that gathering metrics is difficult or inefficient with the current infrastructure. Only the number

of defects can be reported automatically and simple metrics can be gathered by manually calculating or counting them (O6). Senior developer also noted that gathering metrics regarding testing efficiency is difficult, and while some metrics are easily calculated, some other may be very difficult or impossible to acquire (I4). For instance the actual time of executing a test specification has to be estimated after the execution ends. At one point during the summer, this was noted difficult as the execution may have halted for some reason and resumed later (O6). As testing process improvements require more complicated metrics than testing progress or defect status (e.g. Eldth et al. 2010; Chen et al. 2004), and the organization is not gathering them, it may lose valuable information when improving their testing procedures. Other valuable information about testing may be invisible, as well, like what are the operating systems or configurations that are used in testing (O6). Without this information, it may be hard to know which operation systems are not tested and which configurations still need verifications.

With the current system, testing team has difficulties acquiring the information to monitor testing effort that a dedicated test management tool would provide. While test management tool should allow the users to monitor current status of testing activities, such as how many test cases have been run, passed and failed (Eldth et al. 2010), it cannot be done with current system without considerable amount of manual labor. Counting the number of run and failed test cases has been seen exhausting and is prone to errors (O6). The only way to monitor the status is by knowing how many test specifications are executed daily. Because of this, currently the status of test execution is known only vaguely and through communication. (I3) From the process mapping done in the previous section, it can also be seen that the progress is not shown inside the document specification. In other words no one except the tester knows how many test cases have been executed inside the test specification. There are test specifications that need multiple days of work, or test specifications that may be executed alongside other test specifications simultaneously (O6). In these situations, the only progress seen after starting the test specification is when all the cases are done.

The inability to gather data from inside the test specifications causes lack of knowledge is everything tested, extra effort, and untraceability. Some information is not available outside the test specification document, for instance the environment and system under test build with which the test cases are executed (O6). Because of this, no one can see what environments are tested most and which need more attention without considerable manual effort. The need for having the number of test cases available has been seen by the test manager, as well. When a customer wanted information about the test procedures, she had to manually count the number of test cases in these test specifications. This took the work effort of couple of days, and the information was quickly out-of-date. (O13) If another customer interested in testing activities needed this information afterwards, most of the work would have to be done again.

The last activity in test management, reporting, did not explicitly appear in the discussions relating to challenges during interviews. However, as reporting support is closely related to metrics gathering and monitoring activities and metrics gathering has been a difficult task for the case organization, having accurate reports during the test execution and after it is problematic for them, as well. The testing team has various automatic reports of found and fixed defects available for them (O6), which help with deciding is the system under test ready to be released. However, no results of how efficient or effective the testing has been are available for the team. Improving the process by using these results may be difficult, as there is no meaningful information about how well the test executions were realized against the plans, only how it was realized. The plans can be found from version control, and it can be seen that the Excel sheet is completely different from the beginning (O6). Comparing just these sheets to each other can be seen difficult, compared to, for example, quantifiable data.

In summary, the difficulties of the case organization stem from the test asset management and are shown in various test management activities. While it was noted that defect tracking is done in sufficient manner, the ways of finding these defects are not maintained as efficiently mainly because of the lack of designated tool support. The team has to work with general Office tools, which are not optimized for the needs of the team. By not monitoring and maintaining the ways to find the defects through test assets, the quality of the product may be insufficiently portrayed especially if the number of test cases increases in the future. Furthermore, as test cases may be regarded as the only way to document the functionality of the product in exact manner, keeping them up to date can be regarded important.

4.3 Compared characteristics of the tools

In order to establish criteria for evaluation, desired features and other considerations are gathered from the testing team, immediate stakeholders, and through analyzing difficulties and testing process. The first aspect, desired features, depicts how a test management tool might solve various problems with the current test management practices. From these considerations, the final criteria and guidelines for evaluation are set for the project.

4.3.1 Solving problems with desired features

The desired features of a test management tool were discussed during the interviews and during the case project. For this study, the desired functionality is introduced and linked to the challenges in current test management practices in order to see how the features of test management tool may facilitate the problems the organization has. This, subsequently, shows a set of features most optimal for the case organization.

When discussing the insufficient management of test assets, several interviewees stated that storing test cases as individual elements in tool would be very important functionality for a new tool. During the project, this approach was seen to have various benefits. One of the interviewees noted that if the test cases were separate elements, employees changing the details of it would be sure that the changes would be updated to every place. He also recognized that in this case following history information about test cases, for example how the steps have evolved, would be less difficult. (I1) Having the test cases as separate elements from test specification, i.e. having them as the smallest elements that can be managed, would help with the problems the case organization is having with test assets. As separate elements, they could log history information regarding the exact test case, not whole test specification. This would make tracing the evolution of a single test case easier.

Additionally of having the test cases as separate elements, having these elements easily connected to other elements, such as feature descriptions, was noted to be helpful for some problems. One of the interviewees recognized this feature to be helpful in seeing if there are enough test cases to cover all the functionalities of the system under test, a feature which was noted to be very important for seeing if there are enough test cases (I4). This means that number of test cases related to one feature could suggest is it sufficiently tested, or is there a need to create new test cases to increase coverage. This probably would be useful especially with new features. During the project it was noted that the structure of relationships between test cases and features under test would be very complex, as current test cases may be related to many features (O6). Hence, having many-to-many relations with elements in the tool would be important to facilitate with this challenge.

It was believed during the project that by having separate test case elements with relations to features, identifying and reducing the number of duplicate test cases would be easier (O6), which was identified as a problem for the organization. Removing duplicate test cases and having execution information for test case would diminish the need to retest a case that has already been tested in the testing cycle. This would increase the efficiency of testing as number of test cases executed is reduced, while arguably the effectiveness remains the same.

Planning testing tasks was regarded challenging for the case organization due to lack of precise information regarding actual testing efforts. Both in the interviews and through observation it was noted that the effort estimations are imprecise because of the manual estimation of testing efforts. Having test cases as elements for which the time is added through automatic or manual means could help add precision to the estimations, and subsequently add precision to the planning and scheduling process. However, this requires the test cases to be uniform, as it is observed that in different situations executing the same test cases may take different times (O6). Recording test case specific time ef-

fort would also show if a test case takes too much time. For these test cases manual effort could be lessen by some readily available components, such as ready environment. This would be especially important as the team wants to improve the efficiency of executing test cases, and this functionality may pinpoint the problematic areas.

The case organization was noted to have various challenges in scheduling and monitoring testing tasks. The importance of scheduling and monitoring testing activities in many forms was discussed in the interviews, as well. One of testers would find beneficial to see the complete status of testing activities, for example what is been done at the time and what is the team going to do (I2). Similarly improved scheduling and monitoring was requested by other interviewees (I3; I4). Test manager noted that monitoring functionality would be needed, as currently there is no such system in place in the organization (I3). From the management point of view, reporting of current efforts was regarded as a requirement, too (I4; I5). The capability of seeing what everyone is doing could arguably lessen the need to communicate the current status, and subsequently lessen the possibility of misunderstandings and the need of memorizing the complete status of testing, as it could be visualized in the tool. If the team size grows or test manager wants to control the testing process better, this functionality would diminish various problems.

Improved metrics gathering by tool support would facilitate test process improvement. Test manager had noted that improving testing practices could benefit from getting specific data from the process, such as how many test cases are generated during User story testing, how effective these test cases are in finding defects during summer, how many user stories are tested, and how much time User story testing takes (O13). As the current systems do not support this kind of metrics gathering, this information cannot be utilized in process improvement. To get these measurements, the tool should record data on User story basis, linking test cases specifically to a single User story. Hence, it is not sufficient to have only a repository of test cases, but the test cases should include information about how they were created, and the reports should include a way to visualize the information.

Even though not expressed as a challenge for test execution or test management, a tool may facilitate small parts of the process, making performing testing-related activities smoother. For instance, during the observation period, it was noted that sometimes creating a defect report is a bit laborious (O6), but it was not considered a challenge to the process itself (I1; I2). However, because defect reporting is done daily by the testers and it requires combining information from various places, such as steps from test case, expected and actual results, description, module, build, and other environment information (O6), having functionality to facilitate this process could be regarded as an improvement. If the tool would facilitate this process by adding information straight from test case, for instance the steps, expected results, and from used environment, for instance

operating system, build number, and product language, it would reduce the time the tester needs to use recording defect report. Other way around, a tool could create automatically test cases out of existing defects. Using Excel-based tool for scheduling was seen a challenge in the interviews, as well. Similarly as scheduling and rescheduling testing effort was considered troublesome, improving the way it is done could reduce the need to recheck that the changes were made are correct. These small usability improvements in activities, while not regarded as challenges in current context, may make the activities more enjoyable and quicker.

Relating to usability, integrations between different tools in the organization will help the daily use of the complete system. The possibilities to make customizations in a system and integrations with other systems were raised as a desired functionality in the interviews. Modifiability, or expandability, was noted to be an important criterion in many interviews (I3; I4; I5). If the tool could not be changed, it would be very risky to implement it for the organization (I5), and the organization may not be able change its processes without implementing other tools (I4). Regarding integrations, it was noted that defects and test cases should be integrated in the future system (I1). A possibility to integrate systems was noted in many interviews (I3; I4; I5), but it was also noted that the organization may not utilize those possibilities immediately (I5). However, one interviewee speculated that if the test assets were reasonably managed with a new tool, it may not need massive changes afterwards (I1). On the other hand, one other interviewee thought strongly that the process and means of managing the test assets will always change, and the tool needs to be able to adapt to these changes, as well (I3). Nevertheless, improving the testing or test management practices can be seen easier if the tool allows changes to the processes. Omitting this functionality may have dire consequences in the future: if the test case infrastructure is changed somehow, the team needs find workarounds in order to use the tool, which in hand may produce more challenges than solutions.

As a summary, during the project it was noted that functionality provided by a test management tool could help with current challenges and test management activities in various ways. The solutions described in this section were related to various test management activities and possibilities created by integrations and usability improvements. It could be seen that only by increasing traceability between test cases and features under test could solve many difficulties in the testing process. Increasing the accuracy from the level of test specification to test case level can also be seen a major component for resolving various problems regarding history information of executions and changes. Having these elements as the basic artifacts increases the precision of effort estimations. Understanding that not every problem can be solved with tool support may be regarded important, as well. It should be noted that many difficulties, such as lack of test case integrity, may not be solved by implementing a tool. Instead, including the desired level

of detail for test cases into the testing processes would be more proper action than forcing it with a tool.

4.3.2 Criteria in the case project

The criteria in the case project included both functional and quality characteristics for the tool. Some of the requirements were rather straightforward functional requirements, such as ability to manage test assets, facilitate test management activities, and facilitate manual test execution, and these were partially introduced in the previous section. Additionally quality criteria, such as extensibility in form of integrations to other tools, customizability as how the tool can be adapted to current and future testing processes, cost of the tool, and the effort implementing the tool, were taken into consideration in decision-making in the case project. Every criterion was not explicitly set at the start of the project, but some were observable in the reviews and when investigating the functionality. The criteria hence evolved during the progression of the case project.

Table 4: The summary of criteria and their importance in the case project. The importance of each criterion is depicted with a scale low, medium, and high.

Criterion	Importance
Test asset management	High
Planning and scheduling	High
Monitoring test effort	High
Manual test execution environment	High
Usability	High
Customizability	High
Reporting capabilities	Medium
Extensibility	Medium
Vendor characteristics	Low
Cost	Low

During the project, there were various functional considerations, but not all could be thoroughly investigated during the time of the project. The basis of all criteria was that the current processes in the testing team should not be largely affected by introducing the tool. It was noted many times that the tool should not make the process more difficult in practical sense (I1; O7). So, in a sense, the basis for whole project is to make improvements to test management aspects of testing process without losing the advantages seen in the process. From very early, the evaluated functionality investigation categories were restricted to be logical structure of test assets, test monitoring and scheduling, manual test execution, reporting, integration and customizability. Additionally price and licensing was taken into consideration. Prioritizations were done so that first three were of equal importance, followed by reporting and integration, and lastly

the price. (O7; O8) After the first milestone, time and work effort was taken as one criterion, as well, for both evaluation and project management purposes (O9). As the merit cannot be given to each category in a quantifiable form, there needed to be reasoning when considering whether the features are sufficient or not and which alternative is better than other. The whole project revolved around reviewing and evaluating how the different tools fulfilled the requirements set for each group. The summary of the criteria and their importance is included in Table 4.

The logical structure of test assets were defined as a set of features regarding housing, using, and having connections between different testing related elements. The requirements, test cases, test sets, and test runs should be stored in a sensible manner and can have relationships between them. They should take history data of these elements, as well. It should be easy to see certain information about elements, such as found issues and time efforts. There should be a possibility to divide manual and automatic test cases from each other. Increased traceability was wanted, as well. This would help the users to see why something is tested and when it was tested previously. (O7) Test asset management was major area of investigation, and its importance is seen during reviews and discussions, as it was a significant component in order to get the wanted metrics.

In addition to investigation of the structure, facilitating activities of test management, such as planning, scheduling, monitoring, and reporting testing as well as manual test execution were investigated extensively during the project. In the project, the main feature regarding test scheduling was the possibility to schedule test cases and test sets to different testers for test execution. Monitoring included the possibility to track the progress of test execution in order to make possible changes to the schedule. This was considered one of the most important criteria by the test manager, as currently there really is no capability to monitor and control testing process (I3). Reporting capabilities were taken into evaluation criteria, as well. There were two considerations to this: the tool should include various readily made reports with which the testing can be tracked and the tool should have the possibility to make own reports in some manner. Manual test execution was deemed important as the testers would be using the tool on daily basis, and the execution should not be laborious. The requirements regarding manual test execution are: an easy way for testers to test in practical settings while not adding additional steps to current test execution practices. (O7)

Extensibility capabilities by integrations and customization possibilities were part of wanted functionalities for the new tool. The tool should include interfaces with which the information between different systems could be exchanged. This would include a possibility to export and import data from the tool in some format, such as XML or as Excel spreadsheets. Also programming interfaces were desired for further automation. The systems used by the case organization and wanted to be integrated were automation test execution tool TestComplete, continuous integration software TeamCity, and in-

house built defect tracker tool based on the product M-Files. Some kind of custom integration possibilities were required by the team, as the defect tracker was not used in any other organization and thus could not have ready implementation by commercial tool. (O7) These functionalities were taken into consideration to assure the easy use of the complete system and possibilities to customize it if necessary. Integrations and customization were considered second in importance to test asset management and facilitating activities in testing processes (O8). However, at least customization seemed to have a great impact if the tool in question did not adapt to the current processes, as it was noted that the tool should somehow conform to the current processes (O9). It was noted that extensibility is highly valued, as optimally the system will include all testing-related data in the future (I3). Hence even though these criteria were prioritized lower than the functional criteria, having possibilities to extend and customize the tool were regarded very important.

When asked about the most central evaluation criteria from the interviewees' point of view, usability was noted to be a major criterion. All employees involved in the test execution noted that usability should be a priority in the evaluation criteria. The usability became evident in many forms. One tester wanted the tool to be fast to use (I1), the other wanted the user interface to be easy to use (I2). It was especially noted that creating relationships between testing assets should not be difficult (I2). Test manager stated that usability was the most important evaluation criteria; testers would use the tool every day during summertime testing, and if the tool is not usable enough, the work of the testers would become less pleasant (I3). One interviewee noted that usability of the current system is rather good when executing testing and other tools probably would not match that. Thus he said that it is important that usability does not become much worse than in the current system. (I4) This was discussed in another interview, as well, and there were notions that the usability may become worse when introducing another tool, but for the improvements they would need to be worth it (I2).

Usability was visible discussion topic during the reviews of the case project, as well. It was clear that it had great impact when the functionality was demonstrated, as the actual way of planning, monitoring, and executing test cases were discussed lengthily during the reviews (O7; O13). Especially the way the testers have to do their work with the tool was largely criticized (O13). This may also cause one to question the impact the demonstrator's skill had when showing the functionality to other team members. As the demonstrator had investigated only barely the real ways to use the tool, the usability may be shown a way not most favorable for the tool, and thus seen as a limitation. However, this may have demonstrated how easily the tool can be used by a new user and give some information how the tool is used. For more complex tools, this may not be possible, and large amount of training is needed in order to know how to use it efficiently.

When asked about the price being a criterion in the interviews, it was noted that it naturally was, but in this project not very central one (I5). Test manager did not regard it among the most important ones, either (I3). During the project, it seemed that all the prices of the tools were in affordable range. The final explicit criteria, price and licensing and later time effort, was seen to include all the costs needed to implement the tool with all features, train to use it, and maintain it through estimated use period. The initial time of usage was estimated to be three years, and calculations were based on this assumption. Time and work effort translated into costs by using software development work unit, User story points, as the cost of one unit is known for the case organization. It was also considered that the effort needs to be taken out of possibly value creating and income generating activities (O12). In the end, price-related criteria were taken into discussion when comparing the costs of developing the tool internally to the best commercial tool available. In summary, although taken as an explicit criterion and investigated throughout the project, price was not important criterion. On the other hand, work effort was important when seeing how much time and effort was needed when developing a new tool. Also if there are any major differences how long the tools would have taken to be implemented, they could have had an impact on the decision-making, although not very significantly.

Although not taken as explicit criterion for evaluating test management tools, the characteristics of the vendor and its ability to help in problem situations was observed to become part of the evaluation during the case project. When one of the tools did not work and the solution was not easily resolved by the vendor's support team, one of the team members noted that it did not fare well for the tool (O7). The tool was considered top four tools evaluated, but as one feature could not be evaluated, it was removed from the last round. Vendor characteristics were not taken explicitly into consideration, but situations like this may help or make the tool look worse. With one other tool, the vendor's support staff was noted to have done their jobs quickly, and it was considered a positive thing, but not included in the discussion any further (O9). However, as the vendor characteristics were not taken explicitly into consideration and the impact of the state of support mattered only slightly, it was not regarded an important criterion in the case project.

4.4 Evaluation of the tools against the criteria

Gathering data for evaluation of the tools was done by observing the functionality by using trial versions, gathering information from e-mail conversations and demonstrations by the vendor representatives, and using technical sheets and websites. The actual evaluation was done as a team including test manager, program manager, senior software developer, and the author. To borrow the terms used by Kotler and Keller (2006, pp. 214-215), the team consisted of various users of the product as well as deciders, approvers, and buyers. The author acted both as a user and a kind of gatekeeper, who

filters the information coming to the evaluation meetings. This approach may be regarded to have various limitations. For instance, there is a possibility that the person gathering all the information misses something due to not knowing how the tool works, and the author could show incorrect information to the evaluation team during the reviews. These observation-based limitations are minimized by asking whether there is a problem by the respective vendor. But, as the reviewed literature showed, absolute knowledge of a tool is impossible to obtain, and coping with this kind of incomplete information is part of the process. Furthermore, the observed functionality may not be regarded absolute truth, but it can be regarded true in the project's context.

Although the progression of the case project is not part of the scope of this research, it is important to describe it in abstract level. The project for gathering information and evaluating different test management tools in the case organization progressed iteratively in phases, in similar manner as the evaluation processes recognized in the literature review. At the beginning of the project all the tools were investigated to get a general idea of the features and possible limitations that would rule them out from further evaluation easily. Every iteration ended with a review of results and some of the candidates were taken out of the list of considered tools based on the findings. Finally, there was one tool, which based on the established criteria was considered the best of the commercial alternatives. This tool was compared with the internally developed solution. Because of the iterative and excluding nature of the evaluations, not all tools were evaluated to the same extent. Hence it is a natural way to describe the evaluations in the realized phases. The initial criteria evolved a little through the case project, but the desired functionalities remained throughout the whole project.

4.4.1 Exploring features of selected test management tools

Like many evaluation processes described in the literature review, the first phase in case project included initial investigation of the alternatives to rule out candidates that did not have the desired functionality. Although many of the tools may have had some feature to fulfill the needs of the organization, the specific implementations of them were considered insufficient. In addition to these tools a tool called VersionOne was taken into consideration at the start of the project, and thus not included in these evaluations. However, the tool seemed to require also migrating from the old software development tools to work effectively. Because this was not wanted, the tool was omitted from comparison very early. The most important field notes gathered were the feature notions (O8), which were gathered by trying the trial versions of the tools, and meeting notes (O7), which gathered the comments of these functionalities. The evaluation was done by using the vendor webpages to acquire the main features and using trial versions of the tool installed in virtual machines.

The need for improved test asset management was considered one of the main factors to start look for a new test management tool for the case organization. All the alternative tools provided functionality for test assets to be housed and managed. There were different approaches to this, but overall the structures were very similar. All the tools had the element requirement as a central one. The tools also include the basic elements for test and defect management: test cases, some sort of element for grouping test cases, such as test container in SilkCentral, test suite in QMetry, test set in QAComplete and folders in XStudio, Zephyr and TestWave, and defects records. All except TestWave and Zephyr seemed to include configuration information for testing. (O7) It was also noted that all the tools provided means to have relationships between different test assets. In SilkCentral, QMetry, XStudio, and TestWave these relationships were noted to be seen clearly in the user interface, while QAComplete and Zephyr was regarded having unclear relationships as they were not visible from the user interface without entering a view with test case specific information. (O8)

A feature for monitoring and scheduling testing effort was investigated in each tool in order to validate that the tools really have the functionality to assign tests to users and seeing the progress of testing. SilkCentral had execution plans where test cases and testers for each could be assigned. The feature also included the possibility to see how much resources, or work hours, were invested in the execution plan and showed did the testers have enough time to execute all the cases. The tool also had various views with which the progress either by requirement or execution plan basis could be monitored. With these observations, it was noted that scheduling and monitoring functionalities were included rather well in the tool. QAComplete and QMetry had implemented scheduling in the form of tasks; the testing tasks was given to testers and linked to test suites or test cases. QMetry did not seem to have a way to monitor the testing in any reasonable manner, while QAComplete included Gantt diagrams with progress in percentual form to see the progress of testing. (O8) Based on these notions, QMetry was not deemed to have sufficient monitoring capabilities and less than desired scheduling capabilities, while QAComplete had sufficient capabilities to do those (O7). Zephyr included scheduling capabilities by using elements test cycle and test phase, for which testers could be assigned. Monitoring the progress of each test phase or test cycle, mainly by seeing how many test cases were passed, failed, and run, was done with reports. It was noted that the tool did not offer an overview how far the testers were in their testing tasks, but the functionality was deemed sufficient. Scheduling with XStudio was done with test campaigns, for which singular test cases or test case folders were assigned to. This test campaign was instanced when it was wanted to be tested, and the tester was assigned to this instanced element. These elements could be tracked, and the progress shown in a reasonable way with diagrams. Scheduling these campaigns did not seem to be included in the tool. Finally, TestWave did not seem to have any test monitoring or scheduling capabilities, making it more of an test case repository than planning tool. (O8)

The possibility of executing manual test cases was a desired feature in test management tool. All the tools seemed to have means to record information how the test cases are executed, some of which seen more user friendly than others. SilkCentral, QAComplete, QMetry, and TestWave use separate manual test execution view, in which an individual test case and all the steps for it are shown. (O8) The scope of one test case was noted to be insufficient, as the tester cannot see the context of one test case like in the current system (O7). Zephyr had more straight-forward approach to manual execution: the test cases were listed in the interface; steps were got when the case was focused, and the status could be set with drop-down list. This approach was noted to be rather efficient and similar to the test execution process of the case organization. Unfortunately the installation of XStudio was corrupted in such a way that test execution could not be tried and the solution to it was not found in timely manner. From the webpages it was seen that it had similar separate test execution view than the other tools, though. (O8) Even though the tools had their limitations with test execution, they all seemed to be acceptable, as none of the tools were excluded because of it. XCode, however, was not seen positively by the team members because of its lack of functionality in test environment. (O7)

Reporting capabilities of the examined tools were inspected briefly to validate that the tools indeed had the capability to show reports and provide the possibility to add custom reports afterwards. The number of readily made reports was taken into consideration, as well. All the tools seemed to have some kind of reporting capability, though the extensity of it varied between tools. TestWave and XStudio had only simple reporting capabilities with few readily created reports and no possibility to have custom test reports. XStudio, however, did seem to have SQL custom reports in future releases. The tested version of Zephyr did also seem to have only simple reports out-of-the-box, but offered possibility to create own reports. QMetry, SilkCentral, and QAComplete had various readily made reports and included the feature of creating own reports. It was noted that especially SilkCentral and QAComplete had large number of reports readily in the tool (O8), although it was also noted that not all the reports may be used in the case organization, they may not be useful. SilkCentral was also seen to have the best capabilities to customize reports with variety of parameters, such as time range or types of test cases (O8). The minimum reporting capabilities seemed to be the fact that there was a possibility to create test reports for own use, and thus TestWave and XStudio were regarded not to have sufficient functionality for it. (O7)

Integration and customizability capabilities of the tools were mostly investigated by using the webpages or manuals of the tools. All the tools seemed to advertise having Web Services application programming interface (API), which enables communication between tools programmatically through the World Wide Web. XStudio additionally included Software Development Kit (SDK) to create plugins for the tool. (O8) All the

tools hence had the possibility to be extended by the case organization if it wanted to do so. However, the extensibility of APIs was not investigated at all. QAComplete seemed to advertise having integrations with both TestComplete and TeamCity, while Zephyr, XStudio, and QMetry had only readily made integrations with TestComplete. SilkCentral and TestWave were noted to have no readily made integrations to these two tools. (O8)

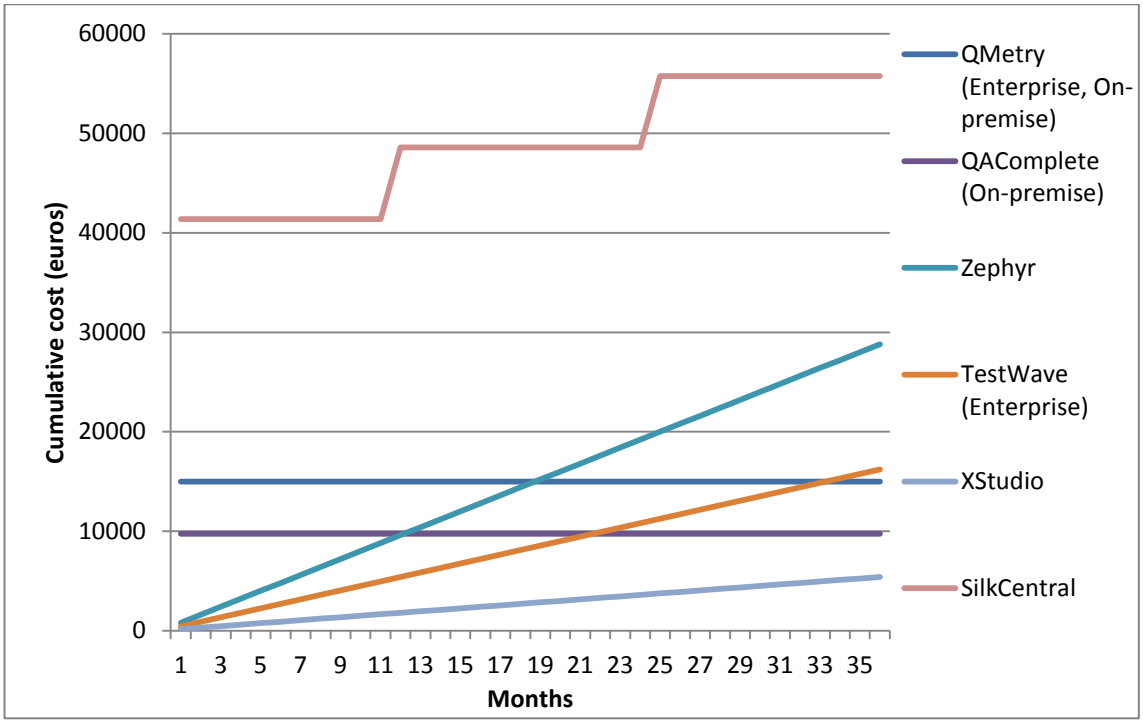


Figure 11: The cumulative costs of each tool in three years' time (O8).

Final consideration for the different tools was pricing and license: how much they cost and what kind of licensing they are offering. The results are gathered in Figure 12. It was noted that SilkCentral was notably the most expensive solution, costing almost twice the amount the next expensive option, Zephyr, would have cost in three years' time. The pricing of other tools varied a fair amount, as well, from the lowest of 5000 euros to 30,000 euros. (O8) The costs were not a factor for excluding any tool at that point, but rather an indication how the costs could evolve over time (O7).

Table 5: Subjective scoring for each tool in the first phase (O7). The unweighted score is inside brackets, while the weighed grade next to it without brackets.

	TestWave	QMetry	SilkCentral	QAComplete	XStudio	Zephyr
Structure	2 (2)	4 (4)	4 (4)	3 (3)	3 (3)	2 (2)
Scheduling and monitoring	0 (0)	1 (1)	5 (5)	2 (2)	2 (2)	3 (3)
Manual test execution	3 (3)	2 (2)	3 (3)	3 (3)	-	3 (3)
Reporting	0,5 (1)	1,5 (3)	2,5 (5)	1,5 (3)	1,5 (3)	1 (2)
Integration	1 (2)	1,5 (3)	1,5 (3)	2 (4)	1,5 (3)	2 (4)
Licensing	1 (4)	0,75 (3)	0 (0)	1 (4)	1,25 (5)	0,5 (2)
Score	7,5	10,75	16	12,5	9,25	11,5

For a basis of discussion regarding which tools were better than others as well as for visualization purposes, the six tools were graded by the researcher with a weighing technique similar to the ones suggested by the reviewed literature. The grade ranging from 0 to 5 was given for each tool by the merit they demonstrated during the trial period and based on secondary data. These grades were weighted by the relative importance of the feature. Structure, scheduling and monitoring, and manual test execution were weighted with 1, reporting and integration with 0.5, and licensing with 0.25. These grades are summarized in Table 5.

Table 6: Summary of tools and required features. The tick (x) represents sufficient functionality.

	TestWave	QMetry	SilkCentral	QAComplete	XStudio	Zephyr
Logical structure of assets	x	x	x	x	x	x
Test scheduling and monitoring	Not included	Not sufficient	x	x	x	x
Manual test execution	x	x	x	x	Included, but could not test	x
Reporting	Simple	Various reports. Own reports.	Various reports. Own reports.	Various reports. Own reports.	Simple. Own reports in future.	Simple
Integration and customizability	Provides Web Services API. No ready integrations.	Provides Web Services API. Ready integration with TestComplete.	Provides Web Services API. No ready integrations for TeamCity or TestComplete.	Provides Web Services API. Ready integrations for TestComplete and TeamCity.	Provides Web Services API and SDK. Ready integrations with TestComplete.	Provides Web Services API. Ready integrations with TestComplete.
Price and license	Middle cast	Middle cast	Most expensive	Middle cast	Cheapest	Second most expensive

The summary of each feature and tool is gathered in Table 6. From the table it can be seen that most tools did include all the desired features, but they were implemented differently. The differences were mostly the extent and usability of implemented feature, and these were the basis of comparison. The comparison between them was therefore done in manner of comparing the best implemented features to less good implementations. It can be also noted that even though most tools had some scheduling and monitoring capabilities, they were noted to be less than desired. Scheduling features, in which the testing tasks were given by separate tasks, for example in QMetry and QAComplete, were regarded difficult to use, while the dedicated testing cycles or execution plans were seen better ones.

4.4.2 Further investigation of the tools with desired functionality

After the first phase QAComplete, Zephyr, and SilkCentral were chosen to be further inspected against the criteria. In the second phase, these three tools were further investigated with logical structure, scheduling and monitoring, manual execution, reporting, integration possibilities, licensing, and time effort. The tools were tested in more realistic situations in which it could be seen how it would work in the real environment. The main field notes used in this section are the presentation of the three tools (O11) based on various sources and notes from the meeting in which the tools were presented (O12).

Logical structure of test assets
In the second phase, the three tools' implementations of the test asset and their relationships scheme were further investigated, as the structure was seen highly important. For this, the infrastructure of testing assets were reverse engineered by observing how different assets seem to be connected. The different elements in the system were gathered and their relationships were visualized with Unified Model Language (UML) notation, which is commonly used in software development. This was done for two reasons. Firstly, the illustrations can help with communicating how the test element architecture is built in the system as well as point of limitations and possible strong points of the tools. These can be used to compare the asset structure with each other. Secondly, the structure may show examples how the in-house solution could be implemented in efficient manner. As the vendors did not provide the wanted figures themselves, the diagrams were done by observing the functionality of each tool. Computer generated diagrams are included in Appendix F.

Doing the visualizations by using practical settings was found to be beneficial for various reasons. First, it was noted that all the alternatives would require changing how test assets were structured in the organization. All the alternatives would add elements such as requirements, configuration, and execution plan, but also separated elements that existed in the current system but were inside test specification, such as test case and test set (O11). By checking how the structure worked in practical settings, perhaps the most valuable information acquired was the fact that one tool, Zephyr, had serious issues with using same test assets between different releases. It was noted that test cases, for instance, lose all the relationships to requirements because they had to be copied to different release. (O12). SilkCentral and QAComplete seemed to have acceptable structures. SilkCentral had reasonably well structured elements: for instance, requirement elements could have other requirement elements as subrequirements. In QAComplete, requirements could not have subrequirements, but requirements could be structured in different folders. Also the relationships or links between elements in QAComplete were noted to be “unclear” and that internally built solution would have better visualization of relationships. It was discussed that if Zephyr had the observed limitations in its structure, it could not be implemented in the organization. (O12) This was later confirmed by the

vendor (O13), and the tool was ruled out altogether. SilkCentral was seen to have the best structure because of its visualized relationships. QAComplete was noted to have reasonably good structure, but the linkings between assets were seen worse than in SilkCentral. (O12)

Planning, scheduling and monitoring activities

During the meetings, it seemed that overall opinion was that test scheduling was not very well implemented in the commercial alternatives (O12). There seemed to be some functionality for scheduling in different tools, but it was done slightly different ways. In SilkCentral, the scheduling was done with Execution Plans, for which the test cases and testers are assigned to from the graphical user interface. These plans are monitored by dashboards in the tool. (O11) Positive about this approach was that one could see how much time a single Execution plan needed and how much resources has been assigned for it, and the dashboards were noted to be the best monitoring tools among the alternatives (O12). Scheduling in QAComplete is done by using metadata, i.e. creating a new property with date on it, and the planner specifies when a certain test set is to be executed. Monitoring the testing progress was done by creating filters based on this metadata. (O11) These was seen rather bad solutions, as the planner needed to set the time for every test set separately, which would take more time than with the current system the case organization uses (O12). In other words, the tool did not take into account the need for planning and scheduling the testing effort. Finally, Zephyr used Cycle and Phase elements to create schedules, which were visualized similarly as in Gantt-chart, although without visualizing the progress, only the order of test phases and length of them. In each Cycle element there are multiple Phases, which include the test cases that are to be executed and information about which testers executes them (O11). It was noted that rescheduling was difficult due to the fact that every Cycle element needed to be rescheduled one by one if any of the Cycles needed to be rescheduled (O12). This would be hard in the current testing processes in the case organization, as rescheduling is done regularly. Otherwise the planning of testing effort in Zephyr seemed to be sufficient, but not as good as in SilkCentral, in which the test cases could be assigned and reassigned easier with the user interface, and the overall situation could be seen with the built-in dashboards.

Overall, it was noted that the tools did not facilitate the planning activities as much as it was hoped (O12). The means to create schedules in the three evaluated tools were a bit different, but SilkCentral was considered to have the best approach only with minor limitations regarding finding test cases to be planned to be executed (O11). Zephyr did have scheduling capabilities, but those were evaluated not sufficient. The planning and scheduling functionality of QAComplete was at this point deemed insufficient for the needs of the case organization. The dashboards showing the current progress of testing was available in all the tools, but SilkCentral did have the best ways to show them with

dashboards. Zephyr and QAComplete relied more on reports, which did not show the progress as quickly or with similar precision as SilkCentral.

Manual test execution

The manual test execution with the tools was discussed in more detail in the second phase, as the team wanted to make sure that manual execution is not difficult and does not provide too many additional steps. The main two elements regarding this were how the user sees his or her scheduled test cases and how the actual manual test execution is carried out by the tester. SilkCentral showed the test cases set for a certain user in a very efficient manner at the front page, and the tester could start testing from the schedule very quickly (O11). The manual execution itself was considered to make the process more difficult, as only a single case was shown for the tester. It was noted that this could be difficult for the tester, as usually the process requires knowing the context, i.e. what other test cases are executed. On the other hand, one team member stated that it might not be a great problem, as one can check the context some other way without much effort. (O12) QAComplete did not provide similar ways to know what test cases were assigned to a specific user for specific date out-of-the-box. When asked from the vendor the right way to see what test cases are to be executed, the answer was that one needs to specify a filter to the listing so that a certain tester sees only his or her test cases (O11). This customization made the test cases visible for the user, but the approach from SilkCentral was seen better because it was easier to access (O12). The actual execution was done rather similarly than in SilkCentral; single test case was shown for the tester. The difference was that the context was shown before the execution was started. Manual execution by Zephyr was seen the best among all the tools. The user could see only the test cases set for him or her under one view, and executing them was done by selecting a state for a test case (O11). This was seen especially good as the tester sees all the test cases set for him in listing so the context becomes clear (O12).

With the current test process, build information was usually added to set of test cases when the tester starts to execute the tests. It was seen that the tools could include customized information, but this was limited to the test case elements, not test runs. Having the build information as configurations was noted to be difficult, and this was identified as a major limitation in all tools. (O12) Not having the build information in test cases was against the current testing processes, and by implementing the tool one would need to change the processes to exclude the build information from the assets, which would not be acceptable. The need for build information was seen important, as by having it included in testing assures that certain test cases have been run after a certain build (O12). For instance, if there was a major fix in certain build, test manager could not see if all the relevant test cases have been run with that or any later build.

Time Effort

In addition to the categories inspected in previous phase, time effort needed to get the tool implemented was also investigated. Most of the time was considered to go to exporting the current elements, such as the contents of test specifications, to a certain format that could be used by the tool and importing the exported information to the tool. This was due to the fact that the case organization had considerable amount of test cases, over 10,000. All the tools could import data from comma separated values (CSV) file or Excel-file. An exporter tool was needed to automate this process, but all the alternatives were considered to need the same amount of work. Although all the tools could import most of the data, they had differences how the imported data was structured in the tool after importing process. SilkCentral was noted to produce a ready structure straight from the file, with no need to manually create any elements before the exporting. QAComplete, on the other hand, needed a folder structure in place in the tool if one wants to import the data to specific folders. If the data was imported without the folder structure in place, all the cases would have to be moved manually to their right places. Both options for the tool were considered to require rather a lot manual effort. Zephyr was seen to have the worst solution, as all the test cases were imported into one folder, and every test case had to be migrated manually from that folder to the wanted folder. Doing this with only few test cases was seen laborious and took a great amount of time, and doing this for all the existing test cases would require enormous effort compared to the other tools. (O11)

Overall, based on the demonstrations with the alternatives, SilkCentral was seen the best alternative of the three (O12). Although SilkCentral had various advantages over the other options making it the best of the commercial products, reason for excluding the others from further investigation was simply because of the central limitations they had. Zephyr revealed serious limitations in its structure, which made it unusable in situations that had several releases like in the case organization. Team members were highly disappointed in the scheduling and monitoring capabilities of QAComplete, as scheduling would still have to be done with other tools, such as Excel. Because SilkCentral worked the best for current processes and included most desired functionality, it was seen a viable commercial option for implementation.

4.4.3 Outlining the internally developed solution

Based on the evaluation in previous phases, SilkCentral was thought to be the best tool among the commercial tools against the criteria (O14). As the price was not a great factor when comparing commercial tools, it was not considered a great advantage or issue for any of the tools. Hence, it is understandable that the priciest SilkCentral with large amount of features and great user interface had the best place among the commercial test management tools. The potential internally developed solution based on the product M-Files was compared against the SilkCentral for various reasons. The team wanted to know how an internal tool would fare against the commercial tool and what effort and

costs would be required to get the tool developed. From these two tools, the team would decide which one would be selected to be implemented. There were three considerations that were taken into account: what readily made components there are, what features needs to be developed to get similar desired functionalities as the commercial tool, and how much effort would developing these functionalities take.

Firstly, all the readily available components were gathered in order to see what could be used for the internal solution. The basis for internal developed solution was the case organization's own product, M-Files. Starting the development from nothing would likely have increased radically the needed work effort and probably made the internally developed alternative unfeasible. With the product as a basis, it would be easier to implement wanted functionalities without major shortcomings. Other available components were applications for the product M-Files. During the project, one team member noted that already developed Gantt chart for M-Files could be used for monitoring (O12). This idea was probably inspired by the monitoring methods of QAComplete. Investigation of the Gantt chart application suggested that with minor additions to the structure, it could show the progress of testing effort rather well (O14). For scheduling, however, it could not be used as easily as hoped. Even though the application allowed the user to schedule and reschedule testing assets to testers, it had usability problems as the dates had to be given for each element separately, and rescheduling needed to be done for each element, as well. In the end, it was noted that the Gantt chart would be used for monitoring purposes, but at least at the beginning scheduling would be done in some other manner. (O14)

Secondly, all the wanted functionalities in internally developed tool were set for the internal solution. By comparing the commercial tools, the wanted functionalities were set to basic requirements and extended functionality. The basic requirements were considered as the basic functionality that was needed to start using the system, while extended functionality can be added later without losing any data in between. The former included the functionality to get the system to gather history data, and it was divided into two elements: implementation of the structure of test assets, i.e. test cases, test sets, and test configuration with M-Files and manual execution improvement by programming a testing environment. This was based mostly on the investigation of the commercial test management tools (O14). Extended functionality would gather the features that were not needed at first. The needed features were set to be integrations to other systems, scheduling capabilities, monitoring functionality with the Gantt chart application, reporting capability, and the actual reports.

As the third consideration, the effort for the wanted functionalities was estimated in order to understand how feasible developing the tool internally is. The structure was noted to be rather easy to construct with M-Files, and the manual test execution environment could be added with small effort. The senior developer, who was part of the

evaluation team, estimated the effort needed to develop all the desired features with User story points (O10). A User story point is a basic unit in the organization for work effort. The costs were estimated by the basic cost for a User story point, which was used internally for development. The efforts and estimated costs of the basic requirements and the extended functionality are gathered in Table 7. It can be noted that basic requirements, which are needed to get the tool in use, can be implanted rather easily and quickly. The other features on the other hand, especially integrations, would need considerable effort because the programming interfaces of all the tools meant to be integrated were unknown for the developer, and he would need to investigate them thoroughly.

Table 7: The functionality of internally developed solution with estimated work effort and costs (O10).

	Estimated User story points	Estimated cost (euros)
Basic requirements	**3**	**4800**
Structure and manual execution improvement	3	4800
Extended functionality	**11-14**	**17,600-22,400**
Integrations to other systems	3 (TeamCity) + 5 (TestComplete) + 1 (Tracker) = 9	14,400
Scheduling and monitoring	1-2	1600-3200
Reporting	1-3	1600-4800
Overall	**14-17**	**22,400-27,200**

Comparing to the costs of commercial tools, the costs of internally developed tool are considerably smaller than with SilkCentral, which was estimated to cost nearly 55,500 euros in three years' time with similar integrations. As comparison, the cost of internal solution would be slightly over the one of QAComplete's (over 22,500 euros) but less than with Zephyr (41,600 euros). This was deemed a very reasonable effort and cost by the deciders (O14). Price-wise all tools except SilkCentral were rather equally matched, while the priciest option, SilkCentral, was over double the cost of the internally developed solution. Having this in mind, the value over the cost should be considered when making the buy or build decision. If the deciders think that the internal solution could meet the offerings of SilkCentral that would be used by the team, the cost could be a decisive factor. The commercial tool probably does have numerous features that may justify the higher cost, but if the organization is not ready to utilize them, they are useless and waste of money.

4.4.4 The conclusion of evaluations

In the end, it was noted that every commercial tool was a letdown, and many team members felt that none of the tools could sufficiently solve the problems the case organization had or facilitate the activities sufficiently (O14). All the commercial alternatives did not sufficiently fulfill all the functional or non-functional criteria. The most significant functional limitation was that the planning of the testing effort in most the tools was not well implemented. Usability of the tools was noted also to be less than desired. SilkCentral had been identified the best of the commercial alternatives during the project, but it did also seem to have few limitations that would require changes in the testing processes. For instance all the tools seemed to have an element called Requirement, which was not present in the present testing processes. This was not seen as a positive feature, as the tools seemed to use it as a very central element. Either the case organization needed to start creating these elements for the sake of the tool or not use them, which would limit the tool's capabilities. (O14) Additionally as the case organization does not have a matured testing process, changes to processes are expected. The tools did not seem to be able to be customized in manner that could easily adapt them to future changes. Instead, they seemed to have been created under certain assumptions on testing practices, which could not be easily changed. While it was discussed that whether these assumed practices could be more efficient for the case organization, it was noted that changes to the tool should not be state how the processes should work (O14). On the other hand, these tools could be used more as test case tools, which would not necessarily require process changes. However, this could be done more easily with the internal solution, as estimated before.

Whichever the tool choice is, the organization needs to change its test asset infrastructure tremendously to increase traceability and efficiency of test asset management. Even if the used tool is changed afterwards to other alternative, these infrastructure changes will not be without benefits. When the changes are done, it would be easier to change for another tool, as all the alternatives seem to have ways of exporting its data somewhere else (O14). Additionally the elements are structured rather similarly in every inspected tool and therefore would not require a large amount of effort to transform the data into a format that other tool requires. This reduces risk that the organization would need to use the selected tool for a long time. The change in decision is, of course, unwanted, but having this in mind there is some room to try an alternative and change it to another if there is a need for it.

Table 8: The comparison between the best commercial tool and internally developed tool against the criteria using a scale satisfactory, fair, and good. Vendor characteristics criterion was omitted from the table because it is not applicable.

Criterion	SilkCentral	Internal tool	Importance
Test asset management	Good	Good	High
Planning and scheduling	Satisfactory	Satisfactory	High
Monitoring test effort	Good	Good	High
Manual test execution environment	Fair	Good	High
Usability	Fair	Fair	High
Customizability	Satisfactory	Good	High
Reporting capabilities	Good	Fair	Medium
Extensibility	Fair	Fair	Medium
Cost	Satisfactory	Fair	Low

The comparison between the best commercial tool and fully developed internal tool is included in Table 8. It can be seen that many functional criteria, such as test asset management, planning, and monitoring were assessed to be rather similar. In other words, the commercial tool did not give an impression that it included such desired functionality that could not be included easily in the internally developed tool. The advantages of internally developed tool with the functional criteria were that they were not hard to develop with the existing components, and by developing them internally the exact manner how the tool functions can be specified. Manual test execution environment, for instance, was not considered great in any of the tools, but with the internally developed tool it can be created exactly how the team wants it. A clear and major difference between the solutions was how the tool could be customized. The commercial tool could be customized to some extent, but internally developed tool could be changed very radically if needed. Less important criteria, i.e. reporting, extensibility, and cost, had major differences, as well. The commercial tool included a great variety of reports out of the box, while the internally developed solution would require effort to create the reports. Furthermore, the commercial tool might offer reports that the team has not thought of. Extensibility through integrations could be seen rather the same, as both offered sufficient APIs and needed all the integrations to be created by the testing team. Finally, although the cost of the tools was not seen an important criterion, the results show that the internally developed tool could be implemented more cheaply than the costly commercial tool, while the price was not low enough to compete with the cheapest test management tools.

Mostly because of the limitations of the commercial tools, the internal solution became more suitable alternative during the project. By evaluating the commercial tools, the team had formed an idea what a suitable test management tool should contain and what it should not. Towards the end, the senior developer of the team, for instance, was convinced that internally developed solution would be better than investigated commercial

tools if given enough resources (O14). This, however, was more based on the lack of desired exact functionalities than the internal solution being better because of better concepts on how testing is done. Evaluating internal solution had some limitations because it was merely a concept next to other alternatives. Some of the criteria could not be realistically evaluated for the internally developed tool, as there was no working prototype to test. One could not, for instance, try how well internally developed solution fared in terms of usability, which was deemed a very important criteria for a tool.

As a summary, there are several points that make the internally developed tool a better alternative than the commercial tool:

1. The internal developed tool is easily built with readily available components. Price-wise it is competitive to the commercial tools, especially against SilkCentral.
2. It is believed that the internally developed tool would better solve the problems the testing team has with test asset management while at the same time it fully adapts into the current testing processes.
3. It evolves hand in hand with the testing processes.

On the other hand, there are still arguments for implementing a commercial tool, which in this case is SilkCentral. The most notable of these are:

1. It would be readily developed solution, which would not require as much investment of resources and more importantly time to be implemented as the internal solution.
2. Support and development is done by external vendor, so that there is no need to invest resources if something breaks.
3. The tool offers numerous functionalities that may be taken into use later without any need to develop them.

With the notions made throughout the case project, it can be argued that internally developed tool would be a better solution if the functional requirements can be matched exactly right. The commercial tools may have included various other even complex functionalities, which the internal solution could not have, but during the case project the organization was not interested in them. The cost difference is great between the best commercial and internally developed tool and as the team thinks that the desired functionalities may be matched or even exceeded with internal solution, the decision may lean towards it.

5 DISCUSSION

Test management is not a term that is widely established in neither in the case organization nor in the literature. The literature views the term as an umbrella term that can be used when management aspects of testing are discussed. The usage seems to be more popular when including discussion with tool support for managing testing activities, and there seems to be numerous commercial tools that brand themselves "test management tools", even though the term “test management” is still vaguely represented in the literature. The discussion of the topics of this study in the literature is rather general, and overall this case study gives some depth of the needs of having a test management tool. The project included also various interesting observations that may not be specifically in the scope of this study. For instance, acknowledging the role of test cases as the main documentation of how certain functionality works or should work may not be part of the scope of this study, but it materialized the need for exact, up-to-date information for test cases, further justifying the need for test management tool. These observations may lead to other questions, such as could test cases be utilized in some manner that everyone would know how the system under test should work, and could this be used in agile methodologies, which do not emphasize on documentation, as a way to know the current functionality.

The case organization is willingly improving their testing practices, and this study helps the organization to decide on a tool that could facilitate test management activities the best. Ultimately, the main purpose of this study was to answer the specified research questions in a way that could be used by both practitioners and academics. In this chapter, the themes of this study are discussed by combining the literature review and empirical data acquired during the case project.

5.1 The need for a test management tool

The case study involved investigation of the need for having a test management tool, and this was done by specifying a research question involving the difficulties in current test management practices. There were numerous difficulties identified in the case study regarding test asset management and test management activities by the team and the author, and some of the difficulties in testing management practices introduced in the literature are evident in the case organization, as well. However, the connections between the difficulties found in the case project and the literature may not be always very clear, and several challenges between sources have many-to-many relationships. In this section, the difficulties identified in different sources are discussed and compared.

Even though the literature was used as an influence to formulate the questions in interviews, the team had already noticed these difficulties without any knowledge about what kind of problems other organizations have encountered. The team had noted that there have been limitations in the current systems for some time and that a new one was needed. This had given the team a reasonable time to notice the specific needs for improvement. Again, because the difficulties in test management practices in this case study focuses on the technical or infrastructural aspects that could be solved with tool support, the scope of the literature is limited similarly to emphasize those elements.

Firstly, the literature suggested that test management tools are used to facilitate many test management activities within a software development project, but there were not many studies that actually showed the benefits of implementing a tool. This study did not show any empirical evidence of having a test management tool, either, as the scope of this study was limited to the definition of the improvement needs and evaluation of alternatives. Furthermore, the solutions achieved by having a test management tool are not empirically witnessed during the case project. Instead, the possible solutions through desired features were discussed during interviews and throughout the project, and no improvement was actually witnessed. This approach was a natural choice in practical sense, because if none of the commercial tools provided the desired features, the organization could have implemented them as features of their own tool. Because of the lack empirical evidence to support that do these solutions really improve the current practices, the significance of these discussions may not be regarded great.

There are similar themes emerging from the literature and the case study. One major component regarding difficulties apparent in both is the need for having more efficient practices, which is not always easily achieved. Testing is extremely expensive, and the organizations in the literature want to make it more efficient while still minimizing the risk that the product would be of less than desired quality for the customers. Ideally there would be no need for testing if there was full confidence that the developed product works in every situation perfectly, but this is not realistic. The need for more efficient practices were not explicitly discussed in the literature, but the problems in schedules (Andersin 2004; Kasurinen et al. 2009) and insufficient investment on it (Parveen et al. 2007; Craig and Jaskiel 2002, p. 9) point imply that. The case organization wanted to minimize the duplicate test cases, diminish extra manual work, and make testing effort more traceable and visible. The need for more efficient practices in competitive environments is not a new observation nor it is limited to software testing, and it was widely recognized in the testing-related literature. Hence these observations only further validate existing knowledge apparent in the literature. But in the end, as software testing is a major way to assure the quality of the software product, making sure that it is well executed should be a priority for any software organization.

Approaching the subject more specifically than with the need to continuously increase efficiency, the difficulties stem from how test asset management is done in organizations. Although the literature notes these kinds of problems relating to test asset management, such as in managing changing testing data (Safana and Ibrahim 2010) and relations between the data (Liu and Robson 1992), this study investigates test management difficulties more extensively than any other literature source, increasing knowledge on the subject in this context. The problems identified in test management in the case organization were caused by the insufficient test asset management, which caused numerous problems in test management activities. The literature does not make similar connections between difficulties in test asset management and test management activities, probably because the terminology may differ and the aspects are not inspected in similar manner. Nevertheless, this observation may be regarded important addition to the current body of knowledge, as it implicates that test asset management has a great impact on test management activities. If the assets are not orderly managed, it may cause problems when performing necessary activities.

The used literature sources described that there is a large amount of continuously changing data that causes difficulties in test management (Lyles 2013; Safana and Ibrahim 2010), problems in finding relations between the data (Liu and Robson 1992), and that testing practices were not sufficiently supported by tools (Safana and Ibrahim 2010; Copeland 2004, p. 4). Similar problems were evident in the case study, as well. The case organization could not gather precise data of its testing processes due to lack of tool support, which in turn made resource allocation an activity based more on hunches or inexact estimates than real data. The scheduling was seen overly inaccurate during the case project and relations between testing data could not be seen. Furthermore, in the product development context the new and the old functionality changed often, which reflected badly on the test assets; the test cases may have been left out-of-date causing inefficiencies in processes later. In evolutionary settings like in this study, these aspects could be argued to be especially important, as the same test cases are executed yearly and there is great potential for metrics gathering. In this kind of environment, however, legacy test cases may cause problems. Increasing number of test cases make knowing what is really relevant very difficult, as there is only hunches of what is affected by the changes and what should be retested. Not having exact information about how long existing test cases took to execute makes planning difficult and inexact, and out-of-date assets cause misunderstandings and extra work. Having literature and the case study complementing each other in this matter, it can be argued that these are common problems that many other organizations may also have. For organizations that experience similar problems, implementing improved test management systems may be warranted.

Although the literature did not distinguish measurement gathering as a major difficulty in test management context, this case study implicates that having insufficient metrics can cause problems for an organization and therefore is an aspect that needs improve-

ment. The literature does suggest that to be effective test management needs various metrics from the testing process (e.g. Craig and Jaskiel 2002, pp. 369; Chen et al. 2004), but it did not note it as a challenge, rather an instrument for improving the way to perform test management activities. The possible uses for metrics are facilitating various tasks, such as estimating schedules, monitoring progress (Craig and Jaskiel 2002), and deciding when to stop testing (Chen et al. 2004). In the case organization, the lack of metrics was an apparent reason for the need of test management tool. Without the metrics, the case organization could not get needed information, including the status of testing progress during the test specification execution phase in summertime. Some metrics were collectable, such as number of test cases and how many of them had failed, but collection of these were laborious. This, in fact, may be considered a difficulty at least in the context of the case organization, and similar problems may appear in similarly matured testing practices. On contrast, as of yet, there is no great body of empirical evidence answering is this really a common problem or does having metrics in test management context produce better plans or lead to better actions. Nevertheless, this shows interesting relation in difficulties between different aspects of test management; difficulties in test asset and resourcing management cause problems in metrics gathering, which was seen to cause problems in performing test management activities. A dedicated system to support test asset management may solve problems in the first one, subsequently helping with the difficulties in the latter ones, as well.

Even though the themes for difficulties in test management practices were similar in the literature and in the case study, the difficulties identified in the case study were not as closely inspected in the literature. The literature did not extensively investigate the difficulties in test management practices like has been done in this study (e.g. Craig and Jaskiel 2002; Parveen et al. 2007), which made the notions in different sources rather general. On contrast, this study deeply inspected the challenges of one company in test management context, which the literature rarely did. Usually the literature investigated the challenges in testing overall (e.g. Ramler 2004) or stated the observed challenges briefly (e.g. Parveen et al. 2007; Safana and Ibrahim 2010), not specifying them on very detailed level. In the end, not one source could be found that had extensively tackled the problems in test management context, and the difficulties are introduced only to provide reasoning for adapting test management tools. Hence this case study increases the understanding of problems revolving around test asset management and test management activities in more detailed manner than in the current academic pieces in the literature. Whether this level of understanding is needed can be questioned, as each organization has a unique environment and having detailed results of one study may not be as beneficial to other organizations as more abstract concepts.

5.2 Developing the criteria for a test management tool

The reviewed literature identified various criteria when evaluating software systems in general and testing tools context. In general, there are numerous aspects that should be taken into consideration, and although every criterion should have a weight in decision-making, gathering every possible kind of data for every tool may not be feasible. Therefore this study focused on the most important criteria specifically in the test management tool scope. The research methodology having influences from action research caused criteria to evolve during the project, and this approach could represent well practical situations; the criteria may not be readily available at the start of the project but instead they are developed during the whole project. The criteria were not based on any literature sources, and although it may be argued that they are influenced by some academic pieces, they were formed without a great amount of background knowledge of what criteria are used in the current literature. Therefore in this section the analysis is done in a manner that compares the results from the case study to the presented literature and tries to find similarities or contradictions between them.

5.2.1 Desired features in a test management tool

Comparing the literature review and the case study, it can be seen that they complement each other with the desired functionalities of a test management tool. In both, the features include improved test asset management and facilitating various test management activities, such as planning, monitoring, and reporting, mainly by gathering metrics automatically (e.g. Majchrzak 2010b; Eldh et al. 2010; Burnstein 2003, pp. 496-497). In fact, all the desired features visualized in Figure 8 were identified features in the case study, as well. Again, the main difference between these two sources is that the case study tackles the desired functionalities in more detailed level, specifying how the tool would work very precisely. This was needed as the internally developed alternative required specific requirements that are used when the system is built. Furthermore, the team had ideas how various test management aspects could be done better in their context, and these ideas were used as a basis for functional criteria. This resulted in a set of requirements that may not have been included in any commercial test management tool. However, having this approach did not limit the alternatives in evaluation much, as the different commercial alternatives, rather surprisingly, did have most of the features that the team had desired.

The literature listed various test asset management (e.g. Majchrzak 2010b; Eldh et al. 2010) and testing resource management (e.g. Eldh et al. 2010) related functionalities, some of which similar to the ones the case organization desired. Some of the specific functionalities are not exactly the same, but in higher level of abstraction they seem to be rather the same. As noted in the literature review, different sources have slightly different implementations of test management tool, but they match rather well in more abstract groups. Similarly it could be seen that by introducing the set of specific re-

quirements, the case study only adds another way to implement a test management tool, which may differ in detailed level but match in more abstract level to the other alternatives. Overall, it can be noted that generally the desired functionality in this study is rather similar to the literature's view of how test management tool should function. It might be argued that the features described in the literature may well be usable in contexts like the case organization. This observation makes communication of test management tools as well as evaluation of them easier, as they can be evaluated based on how these few abstract categories are implemented.

This case study showed that an organization that does not manage its testing assets in sufficient level of detail will have difficulties in performing test management activities. From experiences in the study, one could argue that there is a valid need for gathering and having test case specific information rather than more general test specification information. This is fundamental for gathering metrics, which facilitate various tasks, such as estimating test effort and monitoring it. The literature describing test management does not explicitly offer any certain manner how the test cases should be represented, stored, and managed. For some organizations, having test cases part of test specifications could be a valid and appropriate way to manage test assets. However, if regression testing is as extensive as in the case organization and the number of test cases to be retested is great, the need for having test cases as separate elements could increase. Additionally, the data gathering from a test specification document may be troublesome.

It may be argued that both the case study and the literature agrees that test asset management is mostly about storing assets and finding the right connections between these assets. This is not explicitly expressed in the literature, but looking at the features of a desired test management tool, it can be seen that these functionalities are broadly represented. Without a dedicated test management tool, managing test assets is troublesome, as linking between elements may not be managed well with generic tools such as Microsoft Office products. The relationships between assets were a major problem in the case organization, and having a feature in a tool to create relationships between them was needed. The possible or desired features presented in the literature include similar needs to have separate elements linked to features or other elements, such as requirements (Majchrzak 2010b). Therefore the possibility to have connections between testing elements can be regarded very central functionality in any test management tool.

From the experiences of this study, facilitating test management activities by gathering metrics and creating connections to different systems can be argued to be solutions to various problems in the current practices of the case organization. Automatic metrics gathering decreases manual work effort and errors, providing exact data of the processes. This can be used to plan and monitor testing effort as well as for process improvement. By having the integrations in place, metrics gathering is even more efficient, as

the test management tool can include data from other testing-related sources, such as from automation tools.

5.2.2 Build versus buy

There are two alternatives to get a system that includes desired features: buy one that includes them or build one yourself. There are various differences how build or buy decision is analyzed in the literature and in the case study. The literature shows that various criteria relating to the significance of the product, characteristics of the product and organization as well as situational factors should have an impact on the decision (Ledeen 2011; Daneshgar et al. 2011). On contrast, the case study concentrated only on two of these: on the characteristics of the product, such as how hard the developing is, and situational factors, such as are there enough resources to develop it. Especially the characteristics of the product became very important in the project, as it showed could the internally developed product become as good as the commercial tools. The study did not explicitly investigate the characteristics of the developing organization, such as in-house information systems expertise identified as a criterion by Daneshgar et al. (2011), but rather it was assumed that the organization could develop the tool itself. This assumption could validate the need to look into this aspect as well, although for the case organization it was obvious. If the organization did not have any expertise on developing test management tool, it naturally could not have developed it by itself.

Overall, these inspected aspects are only small subset of the criteria described in the literature. Furthermore, these characteristics were investigated to get knowledge of how feasible the development is, and the potential functionality of the developed tool was compared to the existing functionality of commercial tools. So in fact the comparison was done more in manner that the internally developed tool existed among the other alternatives than giving it a special position. Moreover, it should be noted that strategic significance of the product, which was emphasized in the by both Ledeen (2011) and Daneshgar et al. (2011), did not appear in the discussions in the case project. This further validates the observations made by Daneshgar et al. (2011) that the significance of the product may not be important for small or medium-sized enterprises. On the other hand, the tool was fairly small and easily implemented, and this may have lessened the need to discuss strategic impact, as it may not have had much.

In build versus buy discussion, the reviewed literature omitted an important factor that was considered one that had a great impact in the case study. This factor is readily available components for developing internal solution. In fact, this factor probably made internal development alternative possible, as starting development from nothing would most likely have been considered unfeasible and not considered as an alternative. With the product M-Files, the case organization had already large amount of logic ready that could be used as a basis for a test management tool. Daneshgar et al. (2011) did note that existing infrastructure has an impact on the decision but discussed more on what

other systems are in use for integrations needs, not how they can be utilized in the new system. The literature may not have taken this into account because one could argue that many companies do not have such suitable components readily available. However, many organizations may have ready components without realizing it. Office tools, for instance, are widely used in many organizations and may be used as a basis for a test management tool.

5.2.3 The most important criteria

In the light of the results of this case study, it could be argued that there are some functional and non-functional criteria that are more important than others and that need to be taken into closer inspection when selecting a test management tool within the context of the case organization. The literature sources described a variety of criteria: functional criteria and a great set of non-functional criteria, such as usability, customization, performance, the vendor characteristics, costs, and portability (Jadhay and Sonar 2009; Illes et al. 2005), emphasizing more on extensibility and customization in testing context (e.g. Majchrzak 2012, p. 47; Parveen et al. 2007). This case study mostly investigated the functional criteria and only three non-functional criteria: usability, customizability, and extensibility. Comparing these sources, it can be noted that the literature describes every criteria used in this study in some form. Hence, the criteria found in this study are not new or surprising. On the other hand, there are numerous criteria that were identified in the literature, such as portability, performance, reliability (Jadhay and Sonar 2009), that were not taken into consideration in the case project. The main difference between these two sets is that usually the inspected literature did not prioritize the criteria any way. One could argue that the criteria acknowledged in the literature form a listing of all the possible criteria without any priorities, some of which could be used in certain situations. Therefore the literature and findings in this study do not contradict each other. Instead, the case project suggests a set of criteria that could be most important in this specific context, complementing the views in the current literature.

The functional criteria were extensively investigated in this study, and they were naturally regarded the most important because they solve the problems the organization has. In the case project, the functional criteria were categorized into a few bigger feature groups in order to use them when comparing different tools. These groups included test asset management, which included a variety of functionalities that enhances current test asset infrastructure, and test management activities improvement, which included functionality to schedule, monitor, and report testing effort. In this study, the functionalities matched rather well the needs of the organization, but usually the software products may offer some features that match the needs and a great number of features that the organization may not need at that point. The literature also noted that a tool that includes all the desired functionalities for a test management tool may not exist in commercial markets (Burnstein 2003, pp. 496-497). One could argue that a tool that offers most of the features of test management tool would best fulfill the functional criteria.

Therefore it might be beneficial to set priorities to some features like in the case study, making some features, such as a well-thought solution for test asset management, more important than others. Furthermore, even though a tool may offer a certain feature, the implementation may be done in a manner that is not seen positive. In the end, if too many functional requirements are not fulfilled by a commercial option, one should start looking for alternatives. Indeed, there may be the possibility to internally develop the tool, which may fulfill all the functional criteria.

In addition to functional requirements, there is a variety of non-functional criteria that were important in this case study, most important of which were usability, extensibility, and customizability. Usability, which is also one criterion in the literature (Jadhay and Sonar 2009; Illes et al. 2005), could be considered obviously an important factor because testers and test manager use the tool daily, and a user-unfriendly tool may make the use unpleasant. Usability, however, was not particularly emphasized in the literature in the test management or general contexts. It was noted that test tools could produce problems if they were hard to use (Ng et al. 2004), though, and hence usability could be seen important by some sources. This study showed how the case organization appreciated the usability aspects of the tool, providing more information what may be more important in this context. The importance of integrations and customizability, on the other hand, was a highly discussed topic in the literature by Eldh et al. (2010), Majchrzak (2010b), and Burnstein (2003, pp. 496-497). Hence, the results of this case study further validate the need to take these aspects into consideration when choosing a test management tool. The other non-functional criteria described in the literature but not in the case study may not be unimportant, but it shows that at least in this case they were not considered among the most important and worth to look at.

This case study showed that one cannot overemphasize the importance of having a tool that adapts to the testing team's processes when choosing a test management tool. As Majchrzak (2010a) has stated, usually the test tools are built to be used a certain way, and without changing the testing processes or the tool itself, using the tool may be difficult. Both ways may be troublesome to some organizations. Comparing the varying practices in the literature against the practices of the case organization, it could be seen that testing practices are not well established and there are no common best practices, and it can be argued that some practices may work better for some organizations than others. By investigating different tools, it can be seen that they do require certain sort of process, for instance requirement-driven process with extensive documentation of testing requirements. Although this approach may be recommended by some authors (e.g. Majchrzak 2010b), it was not wanted in the case organization. If a tool requires the use of certain process type, it would not be greatly beneficial to the implementing organization.

To make absolutely sure that the team and testing process can utilize the selected tool, functional criteria became dominant when comparing different tools, while customizability and extensibility were deemed also important because they might complement the lack of desired functionality. To avoid difficulties encountered by Parveen et al. (2007) by having mismatching test management tool and testing process, one of the main criteria during the project became the tool's adaptability to the case organization's processes and procedures. This was very important in the project, as many limitations was found, such as the lack of functionality to include build number when testing. One of the criteria at the very beginning of the project was that the tool must not make the testing practices more difficult than it already is. How this happens, for example, could be that part of the current testing process cannot be done with the chosen tool. For these functional requirements the mapping of testing process will become extremely valuable, and in the case project it was noted that the tools could not adapt to certain aspects of the testing process. This case project, in a way, further validates the claims that extensibility and customizability are major criteria and should be investigated in test management tool context.

Finally, looking at the case project retrospectively, it included a number of functional and quality criteria, and prioritizing them may be difficult, as not all of them were explicitly ranked and they might be ranked differently by different people. It would be beneficial to understand the most important ones among all the criteria. In these inspections, these are divided into functional and non-functional criteria similarly to the literature. Furthermore, the importance of criteria in the case project seemed to be affected by is the solution internally developed or bought commercial product. For instance, implementation effort was more relevant in the internally developed solution, as if the effort was too much, the internal development would not be feasible. On the other hand, the impact of effort for commercial tools was investigated, but it did not weigh much in the decision-making process. Similarly vendor characteristics had a small impact in the process but only for commercial tools, because internal solution did not have a vendor that could be evaluated. Arguably the vendor is the organization itself and the capabilities of the vendor are the organization’s capabilities to support and further develop the tool. However, this was not taken into consideration during the project. Some of the criteria could be said to be equally important regardless of whether the product in question is commercial or internal.

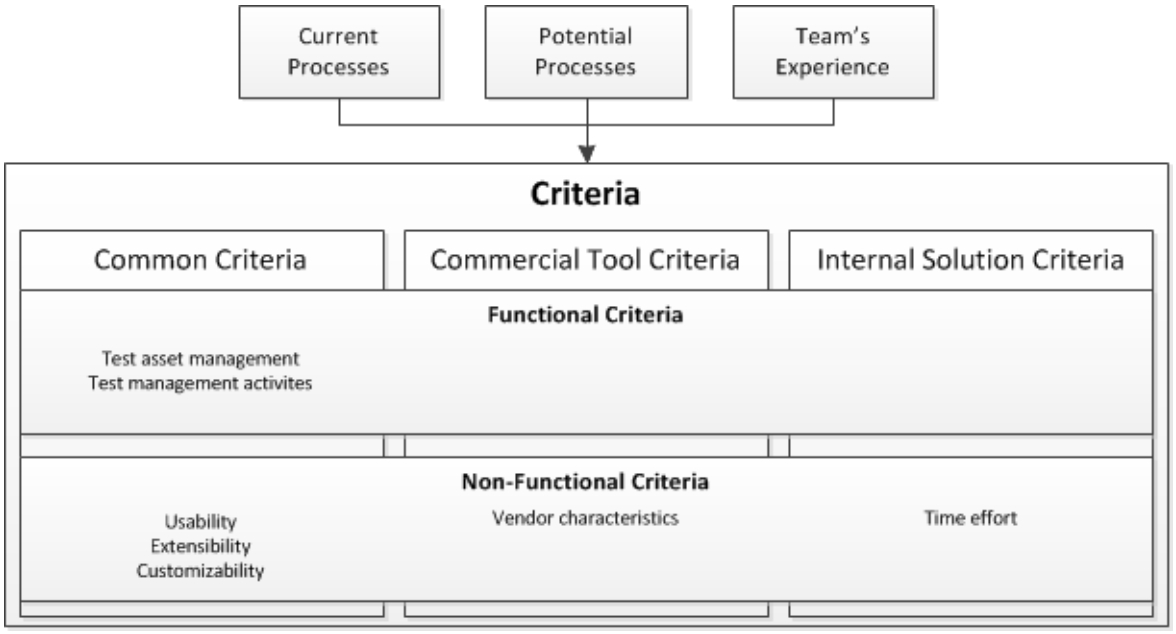


Figure 12: Observed criteria in the case organization.

The most important criteria grouped into common and specialized criteria in the case project and connections from where these are derived are visualized and summarized in Figure 12. The main sources from which these most important criteria were derived were based on the current processes, potential future processes, and the team's experience on what should be included in the tool. Arguably all the criteria were based on these sources. In the project, the most important criteria may be regarded to be the functional criteria, i.e. how the tool should be functioning. On basis of how much effort was put into investigating these features, the most important of these were improved test asset management and easing all the activities in test management, such as planning, scheduling, monitoring, and reporting testing efforts. In addition to the functional considerations, the most important non-functional or quality criteria were apparent in the project. According to many team members, usability was the most prominent of these criteria, and it was extensively investigated. The other noticeable important criteria were extensibility of the tool, customizations to it, and the implementation effort. Extensibility and customizations were among the criteria for easy-of-use and for the future needs. The implementation effort was especially important for the internally developed tool as it was a way to see how long it would take to have the basic functionality done. Less important criteria identified during project were straight purchase costs and vendor characteristics.

5.3 Test management tools against the criteria

Even though the current literature includes some discussion regarding choosing test management tools, there were no articles regarding a process of choosing one among various alternatives in the context of the case organization. Furthermore, there were no literature sources that showed how well different test management tools fulfill a certain

set of criteria. This case study demonstrated how evaluation can be done against a set of criteria in a process, in which the criteria may evolve, filling these gaps in the literature. This section uses the results of this study and inspects how different alternatives fulfilled the separate criteria, increasing information on this aspect.

Firstly, even though the manner how the evaluations were done was not part of the research questions, it can be noted that the case study was in line with the evaluation methods in the literature. The reviewed literature regarding the decision process recognized that to make the decision possible the people making the decision have to identify the needs of the organization, list different alternatives, and finally evaluate the alternatives (e.g. Carney and Wallnau 1998; Khan et al. 1997). The set research questions are in line with the evaluation process described, progressing from specifying what the problems are and places that need improvement, through setting most important criteria to evaluating different alternatives against these criteria. In this study, the two alternatives were to acquire COTS product and to develop the tool internally. This research included in-depth information what test management and a tool for test management could include, partly as a basis for functional requirements for the compared alternatives. The most important criteria were formulated based on the needs of the team and they evolved throughout the process. Finally, the merit was given to the selected tools. Looking the process retrospectively, there were other alternatives too, such as sourcing the solution from external developer or keeping the current test management system in place, but they were not financially, functionality-wise or time-wise realistic ones.

The literature had focused on the software evaluation process in general than in certain context, so this study gives insight how the evaluation can be done in this test management tool context. In the case project, the evaluation was done in a rather “ad hoc” manner than with any systematic models, but the literature did show a similar process of specifying needs, specifying criteria, and assigning merit on the alternatives. Following any process was unintentional, and therefore it can be argued that the steps identified in the literature are common sense. More importantly, this study shows that investigating intensively the steps before actual evaluation can be seen especially important, as the needs of the organization must be evaluated carefully before starting to look for a suitable tool. For instance, if the process for which the tool is implemented is not clear for the deciders, there might be difficulties later as the tools may have some limitations that the current process cannot withstand.

Secondly, this study showed an evaluation of how well functional and non-functional criteria of test management tool are implemented in various alternatives from certain perspective, a subject which is completely omitted from the literature. The literature was not included in this part of the study as largely as in the others, due to the fact that the selected criteria, the alternatives, and the environment were unique for this situation. Moreover, it can be argued that the information produced in this study may not be valid

in the near future, as the tools may evolve, which makes the results of this study quickly out-of-date. More importantly, the case study provides information on how these current specified alternatives fulfill the criteria in a more general manner. This can be used to further develop test management tools, so that the observations made in this study are taken into account.

When looking at the offered features of the test management tools, all the tools were rather similar to each other, and the observed functionality was mostly in line with the functionality the literature described. Every tool had features to manage test assets and logging information about test runs. This seemed to be a basis for all the other functionalities, and, like found in the literature review, very central feature of a test management tool (e.g. Eldh et al. 2010; Majchrzak 2010b). Generally put, the alternatives had rather a well-thought structure for testing assets, including test cases as separate elements and requirement elements that could be linked to test cases. The similarities in general level were also seen in the visualizations of test asset infrastructure; the elements were almost identical and they had relationships with each other. However, certain limitations in the structure caused one tool to be unfeasible. Otherwise test asset management seemed to be reasonably implemented. The internally developed solution described in the study could easily fulfill the needs of the organization regarding test asset management.

Similarly to the test asset management, all the tools had ways to monitor the testing effort with reports or dashboards in some manner. The specific implementation solutions how the monitoring was done differed between tools, and these differences were the basis on ranking them. Overall, although all the commercial tools had a way to monitor testing effort, they were better visualized and accessible by some alternatives, and the implementation of best ones was considered sufficient. Internally developed tool was considered to have had the required monitoring capabilities by using existing components.

Although the alternatives had features that fulfilled most of the functional requirements on abstract level, they lacked in one fundamental test management activity, test effort planning. While some literature sources noted the need for facilitating planning in a test management tool (Eldh et al. 2010; Yoo and Harman 2007), it was one of the less discussed features. Some of the evaluated tools did have functionality to schedule testing efforts, but there was no further functionality to facilitate test planning. For instance, ideally a tool would provide some information about which test cases should be included in regression testing, which was one of the difficulties identified by the case study and the literature, but none of the tools had this functionality. This observation is significant outside this study, as it characterizes a limitation in test management tools. It can be argued that many of the commercial alternatives are limited to test asset management and only to some test management activities, without completely fulfilling the concept that both the literature and the needs for the organization require. Without this function-

ality, organizations may not be able to fully execute all the possible test management activities in one tool. Planning needs to be done with other tools, which can be inefficient, as the information has to be copied between tools somehow, and, like noted by Safana and Ibrahim (2011), it would be most beneficial for an organization to have all the activities done in one environment. In the end, it seems that scheduling or planning of test activities is not as commonly implemented in the commercial testing tools as it would be wanted by the case organization and aforementioned literature sources.

Looking at the non-functional criteria set for the tools, some of them were fulfilled by the alternatives better than others. The case project included usability as one of the most important criteria, but it was not considered to be fulfilled well in many tools. Like described in the study, the usability is important as the testers use the tool daily, and the tools should take this into account. The different commercial tools had graphical interfaces that were deemed acceptable, but the actual test execution interface was not seen in positive light. Internally developed solution would have more possibilities to make the tool as usable as the team wants, although there might be some limitations to that, too, that emerge in further inspection. This type of tool, however, would adapt perfectly to the workflow of the team, which was not the case with the commercial tools.

Both the literature and the case study noted integrations or some other means for extensibility being an important consideration when choosing a test management tool (e.g. Eldh et al. 2010; Majchrzak 2010b), and it was discovered that different commercial tools did provide at least some means for it, fulfilling this criteria. The study noted that there is another aspect to the possible integrations, the number of readily implemented integrations to other systems. Even though every tool had capabilities to be integrated with other systems, they need to be implemented by the organization in order for them to benefit from these integration possibilities. The integration effort was noted to be very great if the organization needed to familiarize themselves with the programmatic interfaces of several tools, taking time and resources. Some tools had some readily implemented integrations to other systems, which reduced the effort considerably. However, all the tools required at least one implementation to integrate the homegrown defect tracker, and many tools did not have readily implemented tools for other commercial systems that were used in the case organization. As every tool provided the means to integrate, it might have been useful to examine how easily the integrations could be done to the system, and use that for evaluation, instead of just the possibility to have integrations as a criterion.

The commercial tools could be adapted to the testing process to some length, which was the basis of how customizability was evaluated in the project. However, some major parts of the process, such as the possibility to include build information in sufficient manner, could not be included in the tool. It should be noted that internally developed tool may be seen to have advantage over the best commercial tool in customizability

and extensibility, as the organization can make various decisions regarding how the tool works and what kind of extensibility possibilities are offered. Furthermore, the organization has a great amount of experience with the product M-Files and integrating it to other tools. As the literature stated that implementations of test management tools may be done subsequently as the testing team evolves (Majchrzak 2010a), internally developed solution could be implemented as far as desired at one point, and as the process evolves, it can be customized and extended any way needed.

As a conclusion, the results of this study showed that different commercial alternatives seemed to fulfill most the functional criteria, and the best commercial alternative could have been a sufficient solution for the case organization. All the other commercial tools had limitations and were not considered to be sufficiently fulfilling the criteria. The commercial tools could have provided additional functionality that the organization had not thought, but these were not seen wanted, and the extra features were left mostly without any investigation. The internally developed tool was seen fulfilling the criteria the best, as it can be adapted to the current processes and needs completely, and there would not be need for any compromises on functionality. This is against the opinion of Craig and Jaskiel (2002), who stated that developing own testing tool is not advisable choice. There may have been, however, problems with some non-functional criteria, such as usability, as there was no way to test how the tool would work on realistic environment because it was not developed yet. Finally, although the implementation without the readily made components was not part of this study, it can be argued that developing all these features from nothing would require a great amount of effort. Possibly not having similar possibilities like the case organization, the development of internal tool may have been unfeasible and not recommended. Fortunately in this case, the readily available tools were sufficient and suitable for developing this kind of tool.

6 CONCLUSIONS

Test management is not well established discipline in the literature, and it consists of various activities that can be done in various ways. The immaturity of the discipline leads to difficulties in performing these activities effectively and efficiently. The case organization of this study had matured from a very small company to a medium one, and it needs to increase its efficiency in testing practices. The testing team of the case organization had witnessed the emergence of the need to improve its processes, and this case study aimed to understand these difficulties ultimately addressing these with a test management tool. When a software organization grows and the testing effort becomes more complex, dedicated tool support is needed to tackle problems caused by the increased complexity.

This study showed that data gathering for a test management tool acquisition is a complex process, including various steps and considerations. In practical settings, the people involved in the process have to evaluate and make decisions based on this data, no matter how imperfect it is. The academic literature may hint what one may want to look for in a test management tool, but specifying what is desired from a tool needs to come from the company, not any external source. The testing process influences greatly how the tool is used. Sometimes commercial alternatives may have limitations that would require changes in the testing practices, and, like in this study, developing the tool internally may become a very suitable choice.

6.1 Answering the research questions

There is an apparent need for a dedicated test management tool in situations like the one with the case organization. Test cases used in the organization are numerous and after each year, the number will increase, making test asset management more problematic. Furthermore, as the organization grows, recoding and monitoring testing efforts in a system may be a good communication tool between test managers and testers as well as between test managers and stakeholders. The discussion in the previous sections leads to an answer to the first research question, *What kinds of difficulties does the case software company have in managing its testing* process? The case study showed various possibly interlacing difficulties, but the answer can be given in a more general sense describing what kind of difficulties a company may have or what causes the difficulties rather than giving a list of difficulties like given in the Results chapter. It was discovered that most of the problems were caused by test asset management, and the difficulties may be characterized by few points relating to management of test assets. The prob-

lems can be seen to be caused by insufficient test asset management, which causes difficulties in the activities in test management. The difficulties may be summarized in two notions in test asset management:

- Insufficient data collection in insufficient level of detail
- Invisibility of relations between continuously evolving elements, such as what is tested and how it is tested

First notion tackles the data collection needed to provide metrics. In the context of this study, this led to having inexact planning, insufficient monitoring capabilities and the inability to improve testing processes by using data. The second notion revolves around the problems caused by not managing changing features and test cases in an appropriate manner. In this study, this caused duplicates, out-of-date test cases, and increasing need of communication. By attending to these difficulties, the case organization can gain better understanding how testing effort is done in the organization and how testing assets are connected to each other. One way to solve these difficulties is by implementing an improved test management systems.

The second research question was formulated as follows: *What features should a tool for test management include so that it will help with the identified difficulties and performing test management activities?* In the study it was noted that most of the difficulties were caused by insufficient test asset management. Therefore the desired features aimed to solve the difficulties especially in test assets. With the observations of the case study, which are supported by the literature, the answer can be stated using three main points with the notions made in the previous sections:

- A tool for test management should include test asset management functionality with which the users can store, maintain, and gather history information of test cases.
- Test cases should be stored as separate elements, so that they may be linked to other elements, such as features of the system under test.
- The tool should include functionality to help with all test management activities, mainly by gathering data of activities and including functionality to actually do different activities within the tool. These include means to plan, monitor, and report testing efforts. Integrations with other tools increase the amount of gathered data as well as help with daily use of the system.

To refer back to the third research question, *What kinds of criteria are the most important in choosing a test management tool*, the proposed answer may be given various ways with the observations done in previous sections. If one emphasizes the background from which the criteria come from, the research question may be formulated also in more general manner. For purposes beyond this case study, this would be more interesting. On the other hand, knowing which of the criteria are the most important in test management tool context would also be interesting for future projects. Answering the

question this manner, however, would arguably be more linked to the context of the case organization and perhaps not generalizable.

Having the discussion in the previous paragraph in mind, the answer to the research question could be formulated as follows: the most important criteria when choosing a test management tool are the ones that solve the identified difficulties and help the tool to adapt to the current and future processes of the organization and takes into account various usability considerations from the team's perspective. In this case, for every alternative these are various functional criteria relating to test asset management and planning, monitoring, and reporting testing effort as well as non-functional criteria, most important of which are usability, customizations, and extensibility. For the internally developed solution the effort that is needed to create the tool was also included as a criterion. Because of the method this study was done, these specific criteria may not be generalized to every situation.

Finally, the last research question was defined as follows: *How do different alternatives fulfill these specified criteria?* Overall, the commercial tools fulfilled most of the functional criteria, including means to test asset management, monitoring and reporting testing, on abstract level, but did not include sufficient functionality for test effort planning. Similarly the non-functional criteria were fulfilled well with extensibility, but usability and customizability of the tools were not seen in high regard. Furthermore, none of the tools could adapt fully to the processes of the case organization. One of the commercial tools fulfilled the criteria best, but it was noted that the internally developed solution could well fulfill the criteria better. Because the internal tool could adapt to the current and expected future needs of the organization the best, it may be regarded the best option at this point.

6.2 Proposed plan of action

The decision regarding which tool should be chosen for implementation is not an easy decision, but the evidence in this study point that the internally developed tool has various advantages over the commercial tool that may lead to developing and implementing it rather than buying any of the commercial ones. SilkCentral, while being a great test management tool, did not provide enough value for its high cost. Internal tool is rather easy to build, and it adapts to the processes exactly without losing to the commercial tools. Based on the information gathered in this study, the proposed action is to develop and implement the internally developed tool, which can be divided into three phases: preparation, usage, and re-evaluation. There are two concurrent sides to this: developing the tool and implementing it. Visualization of these phases is included in Figure 13.

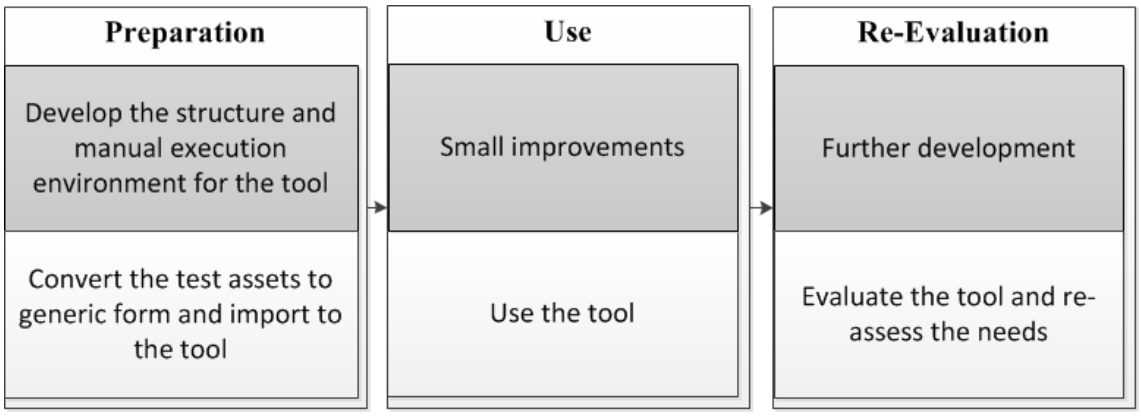


Figure 13: The proposed plan of action for implementing the internally developed tool. Development activities are darkened, while other activities are on light background.

First, the test assets need to be converted in to a format that is accepted by the tool. This would have been mandatory regardless of which tool was chosen, as the organization's goal was to implement one of the tools, and none of them can import the test case information straight from the Word documents. The infrastructure needs to be copied into CSV files, which any tool can read, or alternatively the content can be added via M-Files API. Doing this manually by copying the cases from documents to the tool would be extremely laborious, and it should be helped programmatically with scripting. Even with the automation, someone needs to check manually that every single test case is copied into the new system in a right manner, and this may take some time. Furthermore, the tool must be developed to extent that it can be taken into use to record data. There is no need to actually use the data at this point, as it is more important to start recoding the data for later use. Therefore only the basic requirements of the internally developed tool, i.e. basic structure and manual execution improvements, need to be developed at this point. Starting this phase should not be delayed, as the system needs to start gathering data as soon as possible, and the phase should be completed within a month or two, so the tool can be used throughout the whole next development cycle.

When the tool with the basic functionality is ready to use, it should be used as the only test management system inside the organization. In other words, the old Word based system should not be used anymore, and all the new test cases should be created and run within the new system. With this kind of extensive use, there is a great chance that possible problems with the system emerge. All the problems and improvement proposals should be reported to test team manager or some other responsible person, who records them somewhere for later use. If there are mandatory improvements to structure or manual execution that need to made during this time, they should be done immediately, but preferably not much effort should be put into development at this time, since if there are great problems that cannot be overcome with the chosen tool, this work effort may be considered wasted. This phase will take a whole development cycle, so that the system is tested in both the winter and summer testing.

After the tool has been used for one testing cycle including winter and summer testing phases, the tool and problems should be evaluated. If there is additional time during the summer testing phase, the evaluation phase can be started before it has finished. After this time, there should be enough data to see if the tool has any great problems in it. Before developing more difficult functionalities, such as integrations and reports, the usefulness of the selected tool should be re-assessed. If, for example, the team discovered that there are great limitations to the internally developed tool, there would be no reason to further develop the tool, and other alternatives should be considered. If for some reason the tool cannot be developed usable enough, this might be the time to reconsider commercial tools. This study will then act as a basis for evaluating commercial tools. Finally, only if the tool has gotten the confidence of the team, the tool should be further developed with rest of the functionalities. After this, the tool can be considered to be fully adopted in the organization, but it should be further developed and evaluated after that, too.

6.3 Contributions

Answering the research questions in the organization context was set to help the organization to identify its problems in test management practices and weight on its alternatives when deciding which test management solution they should adopt. The results of this study did exactly that, increasing the organization's knowledge on their limitations and showing how different tools fulfilled their needs. This study limited the alternatives to feasible ones and helped the organization to make the decision. At the end of the project, it was observed that the team was not satisfied with most of the commercial tools, and the internal solution was deemed the most appropriate one. The investigation within this study made this possible. Whether this is the right choice cannot be known before actual implementation and comparing the situation before and after that. Nevertheless, the case organization got its first decision made because of this study.

Although the main purpose of this study was to answer the research questions in order to help the case organization, it contributes to the current body of knowledge in more broad sense, as well. Firstly, the results of this study have contributions to the academic community. The literature review of this study gathered a great amount of information about test management and test management tool. There are no other academic literature pieces that evaluate the current state of these subjects as extensively as this study. Furthermore, the discussion in the study provided information for future research mainly on the difficulties in managing testing and criteria on test management tool context. This study also provided an extensively described case that can be used among other cases to validate various hypotheses regarding the subjects of this study. Secondly, the results have meaning for the practitioners, as well. With the information provided in this study, other organizations may find similarities in difficulties and see if similar solutions may work for them. The most important criteria identified in the study may give

idea for vendors what organizations like the one in this study is looking for in a test management tool and other software organizations a basis to investigate their own criteria.

6.4 Limitations

Through reviewing test management literature it was noted that the subject is not very established in it. The lack of literature providing a solid base for this research caused problems during the project. Related terminology was not common enough, which caused different usages in different literature sources. This caused difficulties to find relative literature, and some aspects may not have been found because wrong terms were used. In the end, the author needed to create the terminology and concepts based on various sources, and use that as a basis for further research. There was no certainty that this basis was correct, which may have caused inaccuracies in the literature review.

The thesis was done in a practical setting, and naturally the sponsor of the project wanted results as soon as possible. The need to have information quickly may have had negative impacts on the research and documentation. As the acquisition project was started immediately when this thesis started, there was not much time to review the literature beforehand. Looking back at the research retrospectively, the thesis would have benefitted from the literature especially with the first research question regarding difficulties in test management practices. The semi-structured interviews could have had more specified questions about the problems identified in the literature, making the research more deductive.

The case study can be regarded largely subjective, which may have implications to reliability and credibility. As a single case study, the results acquired in it may not be generalizable to other organizations, and the implications may not be beneficial in other settings. Based mostly on personal opinions in only one context, the results are applicable in the organization's setting, but they naturally cannot be reproduced in other environments. Furthermore, there is a chance that some aspects may have been overly emphasized by the team in interviews. For instance, some of the problems may have seemed great in interviews although in practice they were not. Similarly some of the difficulties identified in this study may have been highlighted in the observations if they were especially visible in the literature. The observations made by the author were tried to be validated by the members of the team, but this was not always possible.

It should be noted that the data gathered from the tools is not completely accurate. As Carney and Wallnau (1998) had stated, there is "not perfect information" available when choosing a tool, and this was witnessed during and after the project. The data was gathered with technical sheets, through the researcher's observations with trial versions of the tools, and with e-mail conversations with vendor's representatives. This is appar-

ent limitation to any research based on practical settings, but on the other hand depicts the evaluation process realistically. During the evaluations, it was noted that there were various ways to do the data gathering. It was important at the later stages of the evaluations that real-world scenarios were used. However, this was problematic, as creating such an environment was laborious. Additionally the evaluation of usability could have been done better, as the people evaluating the usability could not really use the tool themselves for longer periods of time. Therefore, piloting one or two tools for a short time could have produced better results on many aspects, increasing credibility. In this case it was not possible due to time limitations and lack of resources. Usability was in the end evaluated by the author's personal opinions and with demonstration sessions, in which other people evaluated the usability aspects without really using the tool themselves. Also evaluation of other criteria could have been benefitted from the actual usage of a tool by several team members.

Finally, the evaluations made in this study may also be regarded highly subjective, possibly reducing the credibility of the results. The manner the evaluations were done relied highly on data gathering done by the author, and the evaluations were done in the team based on the demonstrations and observations made based on this data. Although this was a practical way to do it organizations, the credibility of the results may have been compromised as the observations may not have been perfectly accurate or the demonstration may not have shown full functionality. Again, the evaluation with multiple people could have lessened the impact of this, but in practical settings this may not be feasible.

6.5 Implications for further research

This research answered few questions about justification, criteria formulation, and evaluation in test management tool acquisition, but it posed other questions for future research. As stated earlier, this study may be used as a single case in a set of case studies regarding different themes of the study. However, as of yet, the number of these types of studies is very limited, and therefore there needs to be more research to generalize the subjects of this study.

Firstly, the starting point of any acquisition project is justifying building or buying a tool. Starting the acquisition project could be done more easily if there were more empirical evidence on the subject of improving test management and test management tool. Not being well established in the academic literature, test management may be a discipline that is vague and difficult to grasp. Currently, the test management literature mostly consists of best practices, not on any evidence of how beneficial it is for organizations. This study showed that test asset management problems can cause various difficulties in other aspects, but as it was conducted only as a case study, there needs to be more empirical evidence that these problems appear in other contexts, as well. Similarly

several tools have been developed for test management, but there is only little evidence how they enhance performing test management activities. Future research should provide additional information on these aspects, as without good knowledge whether having improved test management benefits the organization, investing money and resources on it may be difficult to justify.

Additionally to producing certain criteria for test management tool context, this study puts forth an interesting question about how the set criteria should be formulated. Rather than providing a set of the most important criteria that may or may not be applicable in other environments, it might be more interesting for future studies to form efficient ways to recognize the most important criteria, as arguably not everything can be evaluated in a feasible manner. Currently, the literature does not specify the most important criteria in the test management context or any means to identify them. This could be beneficial in other projects. Specific functional criteria, for instance, may not be applicable in other projects, as the processes may differ between organizations greatly. Furthermore, providing an evaluation method that gives a weight for each attribute is meaningless if the attributes are not correct or relevant for the organization in the first place.

7 REFERENCES

Abran, A., Bourque, P., Dupuis R., Moore J. W. and Tripp L. L. 2004. Guide to the Software Engineering Body of Knowledge – SWEBOK. 202 pp.

Agarwal, B. B., Tayal, S. P. and Gupta, M. 2010. Software Engineering and testing. Jones & Bartlett Learning, Sudbury, Massachusetts, the United States of America. 516 pp.

Aljahdali, S., Hussain, S. N., Hundewale, N. and Poyil, A. T. 2012. Test Management and Control. Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference, pp. 429-432.

Almog, D. and Heart, T. 2009. What Is a Test Case? Revisiting the Software Test Case Concept. Software Process Improvement. Springer Berlin Heidelberg. pp. 13-31.

Andersin, J. 2004. TPI – a model for Test Process Improvement. Seminar on Quality Models for Software Engineering.

Bertolino, A. 2007. Software Testing Research: Achievements, Challenges, Dreams. 2007 Future of Software Engineering (FOSE '07). IEEE Computer Society. pp. 85-103.

Black, R. 2008. Advanced Software Testing Vol. 2: Guide to the ISTQB Advanced Certification as an Advanced Test Manager. Rocky Nook, Santa Barbara, California, the United States of America. 570 p.

Black, R. 2002. Managing the testing process: Practical Tools and Techniques for Managing Hardware and Software Testing. 2nd edition. Wiley Publishing, Inc. New York, NY, the United States. 500 p.

Brooks, F. P. 1987. No silver bullet-essence and accidents of software engineering. IEEE Computer, Vol. 20, Issue 4. pp. 10-19.

Burnstein, I. 2003. Practical Software Testing: A Process-Oriented Approach. Springer Professional Computing, New York, NY, the United States of America. 709 p.

Carney, D. J. and Wallnau, K. C. 1998. A basis for evaluation of commercial software. Information and Software Technology, Vol. 40, Issue 14. pp. 851-860.

Chen, Y., Probert, R. L. and Robeson, K. 2004. Effective test metrics for test strategy evolution. Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research. pp. 111-123.

Cohen, C. F., Birkin, S. J., Garfield, M. J. and Webb, H. W. 2004. Managing conflict in software testing. Communications of the ACM. Vol. 47, Issue 1. pp. 76-81.

Copeland, L. 2004. A Practitioner's Guide to Software Test Design. Artech House, Norwood, MA, the United States of America. 320 p.

Craig, R. D., Jaskiel, S. P. 2002. Systematic Software Testing. Artech House, Norwood, MA, the United States of America. 536 p.

Davis, C. 2006. Test management best practices. IBM developerWorks [electronic journal]. Referenced on 25.06.2013. Available at: http://www.ibm.com/developerworks/rational/library/06/1107_davis/.

Daneshgar, F., Worasinchai, L. and Low, G. 2011. An Investigation of 'Build vs. Buy' Decision for Software Acquisition in Small to Medium Enterprises. Society of Interdisciplinary Business Research (SIBR). 2011 Conference on Interdisciplinary Business Research. pp. 1741-1750.

Desai, H. D. Test Case Management System (TCMS). 1994. Global Telecommunications Conference. GLOBECOM'94. Communications: The Global Bridge., IEEE. Vol. 3. pp. 1581-1585.

Eldh, S., Brandt, J., Street, M., Hansson, H. and Punnekkat, S. 2010. Towards fully automated test management for large complex systems. Third International Conference on Software Testing, Verification and Validation. pp. 412-420.

Engström, E., Runeson, P. and Skoglund, M. 2010. A systematic review on regression test selection techniques. Information and Software Technology. Vol. 52, Issue 1. pp. 14-30.

Farrell-Vinay, P. 2008. Manage Software Testing. Auerbach Publishers, Boca Raton, FL, the United States of America. 600 p.

Gao, L. 2011. Research on implementation of software test management. 2011 3rd International Conference on Computer Research and Development (ICCRD). Vol. 3. pp. 234-237.

Gilbert, M. R., Shegda, K. M., Chin, K., Tay, G. and Koehler-Kruener, H. 2012. Magic Quadrant for Enterprise Content Management [electronic journal]. Referenced on: 01.08.2013. Limited availability. Available at: http://www.gartner.com/id=2204420

IEEE Computer Society. 2008. IEEE Standard for Software and System Test Documentation. IEEE Standard 829-2008.

Illes, T., Herrmann, A., Paech, B. and Rückert, J. 2005. Criteria for software testing tool evaluation. a task oriented view. Proceedings of the 3rd World Congress for Software Quality. Vol. 2. pp. 213-222.

ISO. 2008. ISO 9000 Introduction and Support Package: Guidance on the Concept and Use of the Process Approach for management systems. Referenced on 05.08.2013. Available at: http://www.iso.org/iso/04_concept_and_use_of_the_process_approach_for_management_systems.pdf

ISTQB Guide. n.d. What is Test management tools? [electronic journal] Referenced on: 01.08.2013. Available at: http://istqbexamcertification.com/what-is-test-management-tools/

Jadhav, A. S. and Sonar, R. M. 2009. Evaluating and selecting software packages: A review. Information and software technology. Vol. 51, Issue 3. pp. 555-563.

Jenkins, N. 2008. A Software Testing Primer. An Introduction to Software testing. [electronic journal] Referenced on: 01.08.2013. Available at: http://www.nickjenkins.net/prose/testingPrimer.pdf

Kasurinen, J., Taipale, O., Smolander, K. 2009. Analysis of Problems in Testing Practices. Software Engineering Conference, 2009. APSEC'09. Asia-Pacific. pp. 309-315.

Khan, M., Ramakrishnan, M. and Lo, B. 1997. Assessment Model for Software Maintenance Tools: A Conceptual Framework. PACIS 1997 Proceedings. Paper 51.

Kotler, P. and Keller. K. 2008. Marketing management. 13th Edition. Pearson Education India. 816 p.

Langer, A. M. 2012. Guide to Software Development: Designing and Managing the Life Cycle. Springer-Verlag, London. 350 p.

Lazic, L. and Mastorakis, N. 2008. Cost effective software test metrics. WSEAS Transactions on Computers, Vol. *7*, Issue 6. pp. 599-619.

Ledeen, K. S. 2011. Build v. Buy: A Decision Paradigm For Information Technology Applications. [electronic journal] Referenced on 01.08.2013. Available at: http://www.nevo.com/our-knowledge/whitepapers/ShowPDFPopup.asp?ID=5

Liu, L. and Robson, D. J. 1992. SEMST - a support environment for the management of software testing. Assessment of Quality Software Development Tools. Proceedings of the Second Symposium on. pp. 11-20.

Louridas, P. (2011). Test Management. IEEE Software. Vol 28, Issue 5. pp. 86-91.

Lyles, M. 2013. A Day in the Life of a Test Manager. Limited availability.

M-Files. 2013. Electronic document management – EDCM and DMS – M-Files. Referenced on 01.08.2013. Available at: http://www.m-files.com/en/.

Majchrzak, T. A. 2010a. Best Practices for the Organizational Implementation of Software Testing. Proceedings of the 43rd Hawaii International Conference on System Sciences.

Majchrzak, T. A. 2010b. Improving the Technical Aspects of Software Testing in Enterprises. International Journal of Advanced Computer Science and Applications, Vol. 1, Issue 4.

Majchrzak, T. A. 2012. Improving Software Testing Technical and Organizational Developments. Springer. 160 p.

McIvor, R. 2005. The Outsourcing Process: Strategies for Evaluation and Management. Cambridge University Press, Cambridge, the United Kingdom. 338 p.

Naik, S. and Tripathy, P. 2011. Software testing and quality assurance: theory and practice. John Wiley & Sons, Hoboken, New Jersey, the United States of America. 616 p.

Neto, C. R. L., de Almeida E. S., and de Lemos Meira, S. R. 2012. A software product lines system test case tool and its initial evaluation. Information Reuse and Integration (IRI).

Ng, S. P., Murnane, T., Reed, K., Grant, D. and Chen, T. Y. 2004. A preliminary survey on software testing practices in Australia. Proceedings of the 2004 Australian Software Engineering Conference (ASWEC'04). pp. 116-125.

Onoma, A. K., Tsai, W. T., Poonawala, M. and Suganuma, H. 1998. Regression testing in an industrial environment. Communications of the ACM. Vol. 41, Issue 5. pp. 81-86.

Oxford Dictionaries. 2013. tool: definition of tool in Oxford dictionary. [www]. Referenced on 01.07.2013. Available at: http://oxforddictionaries.com/definition/english/tool?q=tool.

Parveen, T., Tilley, S. and Gonzalez, G. 2007. A case study in test management. Proceedings of the 45th annual southeast regional conference. ACM.

Poston, R. M. and Sexton, M. P. 1992. Evaluating and selecting testing tools. Software, IEEE. Vol. 9, Issue 3. pp. 33-42.

Ramler, R. 2004. Decision support for test management in iterative and evolutionary development. Proceedings of the 19th IEEE international conference on Automated software engineering. IEEE Computer Society. pp. 406-409.

Safana, A. I. and Ibrahim, S. 2010. Implementing Software Test Management Using SpiraTeam Tool. 2010 Fifth International Conference on Software Engineering Advances (ICSEA). pp. 447-452.

Saunders, M. N. K., Lewis P. and Thornhill A. 2009. Research methods for business students. 5th edition. Pearson Education Limited, Essex, England, the United Kingdom, 614 pp.

Traq Software Ltd. 2011. How to Evaluate Test Management Tools. [electronic journal]. Referenced on 01.07.2013. Available at: http://www.testmanagement.com/resources/eval-testtools.html

White, L. J., Narayanswamy, V., Friedman, T., Kirschenbaum, M., Piwowarski, P. and Oha, M. 1993. Test manager: A regression testing tool. Conference on Software Maintenance. pp. 338-347.

Whittaker, J. A. 2000. What is software testing? And why is it so hard?.Software, IEEE, Vol. 17, Issue 1. pp. 70-79.

Wolfinger, N. H. 2002. On writing fieldnotes: collection strategies and background expectancies. Qualitative Research. April 2002, Vol 2. pp. 85-95.

Yang, H. and Ward, M. 2003. Successful evolution of software systems. Artech House, London, the United Kingdom. 300 p.

Yin, R. K. 2003. Case study Research – Design and Methods. Third Edition. SAGE Publications, Thousand Oaks, California, the United States of America. 312 p.

Yoo, S. and Harman, M. 2012. Regression testing minimization, selection and prioritization: a survey. Software Testing, Verification and Reliability. Vol. 22, Issue 2. pp. 67-120.

8 APPENDICES

Appendix A: The structure of semi-structured interviews
Appendix B: The list of interviews made
Appendix C: The list of observation field notes and secondary data
Appendix D: User story testing process
Appendix E: Test specification testing process
Appendix F: Structure visualizations

Appendix A: The structure of semi-structured interviews

Structure of interviews used in thesis

1. **Information about the interviewee**
    - Name
    - Job description
    - Experience on testing or test management
2. **Tasks done for testing**
    - What testing related tasks do you have in your job description?
        - During summer and winter time?
        - Can you tell me how you do some parts of your work, i.e. designing test cases or scheduling test cases?
        - For testers: how you define the time used? Is it difficult?
    - In your job, is there tasks that are difficult, time consuming, or even useless?
        - Finding testing tasks, defect reports, other required information?
3. **The difficulties in management of testing from the team's perspective**
    - In your organization, what do you see especially good in test management or in testing?
        - Winter time?
        - Summer time?
    - In your organization, what do you see central problems in test management or in testing?
        - Winter time?
        - Summer time?
    - What would you change in the current practices?
4. **The most important criteria for a test management tool among different stakeholders**
    - How would you like tool support help you in your tasks?
        - Make easier? Quicker? Clearer?
    - What functional or quality criteria would you have for the tool?
        - Which situations the tool should help you with?
        - What is essential and what is very important?
        - Are there scenarios that need to be fulfilled?
    - Do you have concerns about the implementation?
    - What would be the most important evaluation criteria for you when choosing a tool?
    - Mitä toivoisit työkalun tarjoavat tulevaisuuden tarpeita varten?
    - What would you hope the tool to help in the future?
        - Would there be many changes after the implementation?
        - For example: Changeability? What kind of change? Increased automation support?

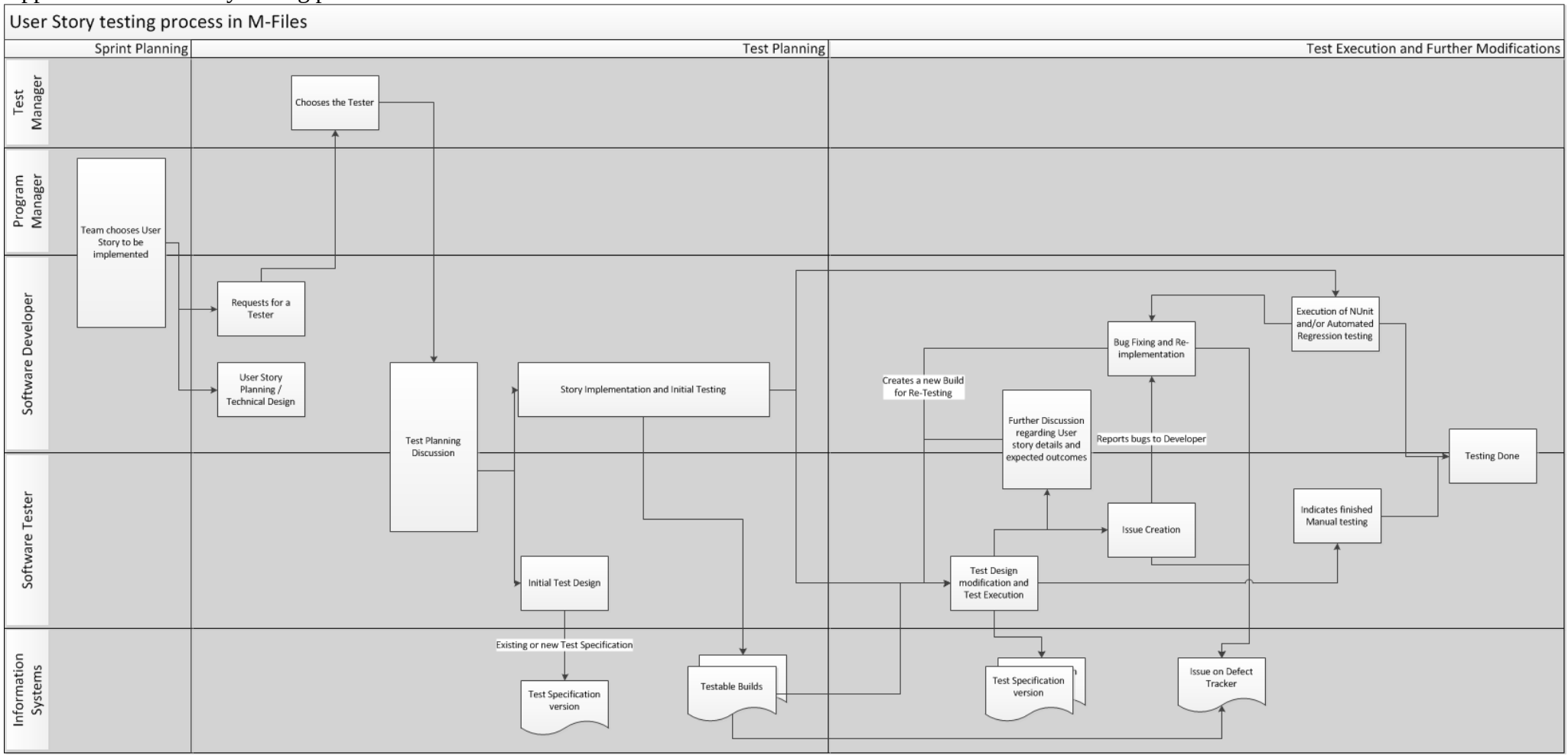
Appendix B: The list of interviews conducted

Notation	Date	Interviewee	Notes
I1	26.4.2013	Software Tester	
I2	29.4.2013	Software Tester	
I3	6.5.2013	Test Manager	
I4	6.6.2013	Senior Software Developer	First theme was skipped because the interviewee was not part of testing team at the time. Other themes were discussed, though, as the interviewee had notable experience in testing and test management in the organization.
I5	19.6.2013	Director of Research and Development	

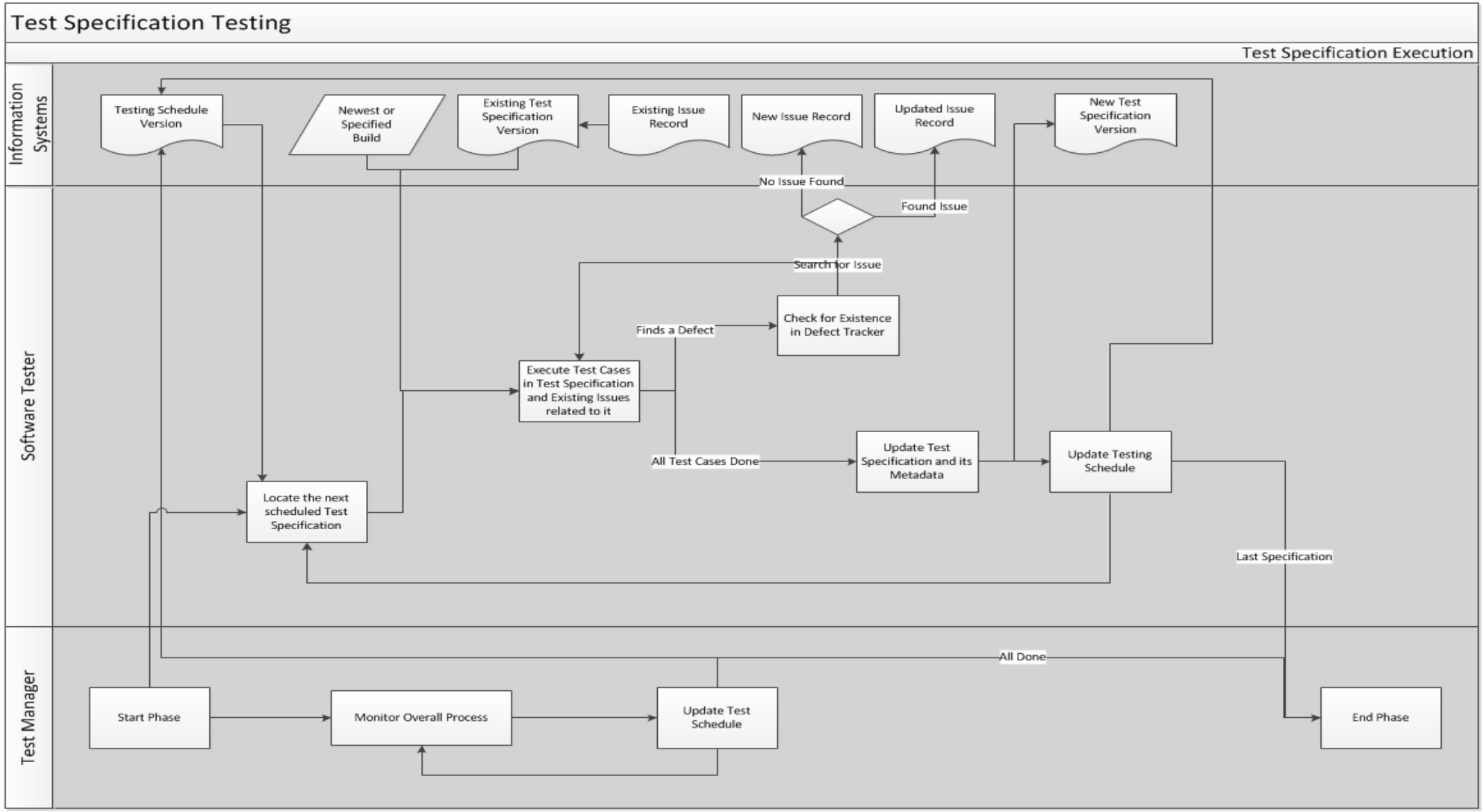
Appendix C: The list of secondary data and observation field notes

Notation	Time	Note name	Description
O1		Financial statement of M-Files for 2012	
O2		Testing process in M-Files	Testing process mapped in the organization based on various sources
O3		Definition of done document	A checklist for specifying is a User story done
O4		Timetable for system and integration testing, M-Files 10	
O5		Test plan for M-Files 10.0	
O6	04.03.2013-05.07.2013	Difficulties in test management observation notes	
O7	06.02.2013	Meeting notes of the first milestone	Prioritization of features and meeting notes of the first milestone. Tools included: SilkCentral, QMetry, VersionOne, QAComplete, Zephyr, TestWave, XStudio
O8	01.02.2013	Tool comparison sheet	Included tools: SilkCentral, QMetry, VersionOne, QAComplete, Zephyr, TestWave, XStudio
O9	13.03.2013	Tool comparison sheet	Included tools: SilkCentral, Zephyr, QAComplete
O10	28.02.2013	Developer discussion	Regarding the needed effort to create the tool in-house
O11	11.03.2013	Presentation of the second milestone	Tools included: SilkCentral, Zephyr, QAComplete
O12	11.03.2013	Meeting notes of the second milestone	
O13	04.03.2013-03.06.2013	Discussion notes with Test manager	
O14	04.04.2013	Meeting notes of the third milestone	The last meeting to compare the best commercial and internally developed tool.

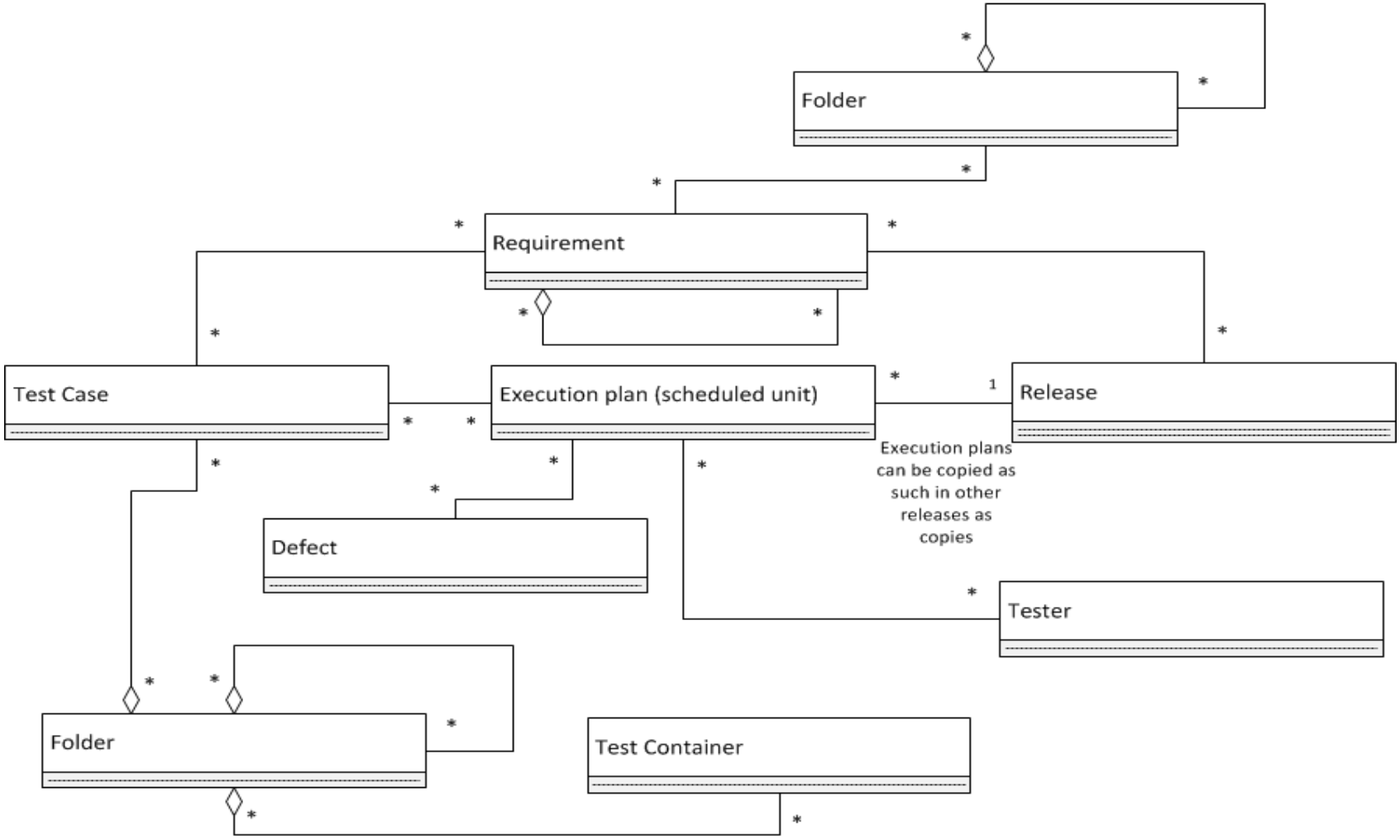
Appendix D: User story testing process



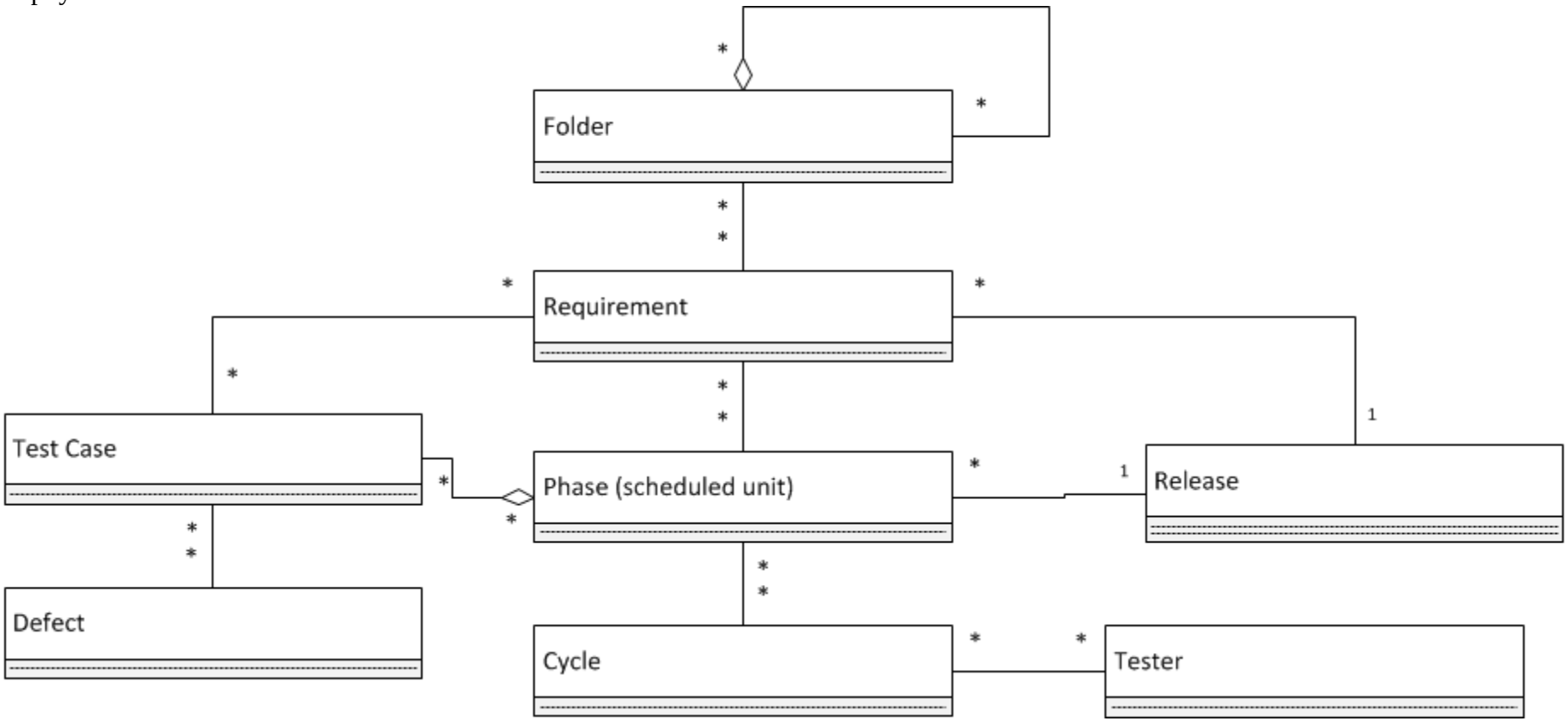
Appendix E: Test specification testing process



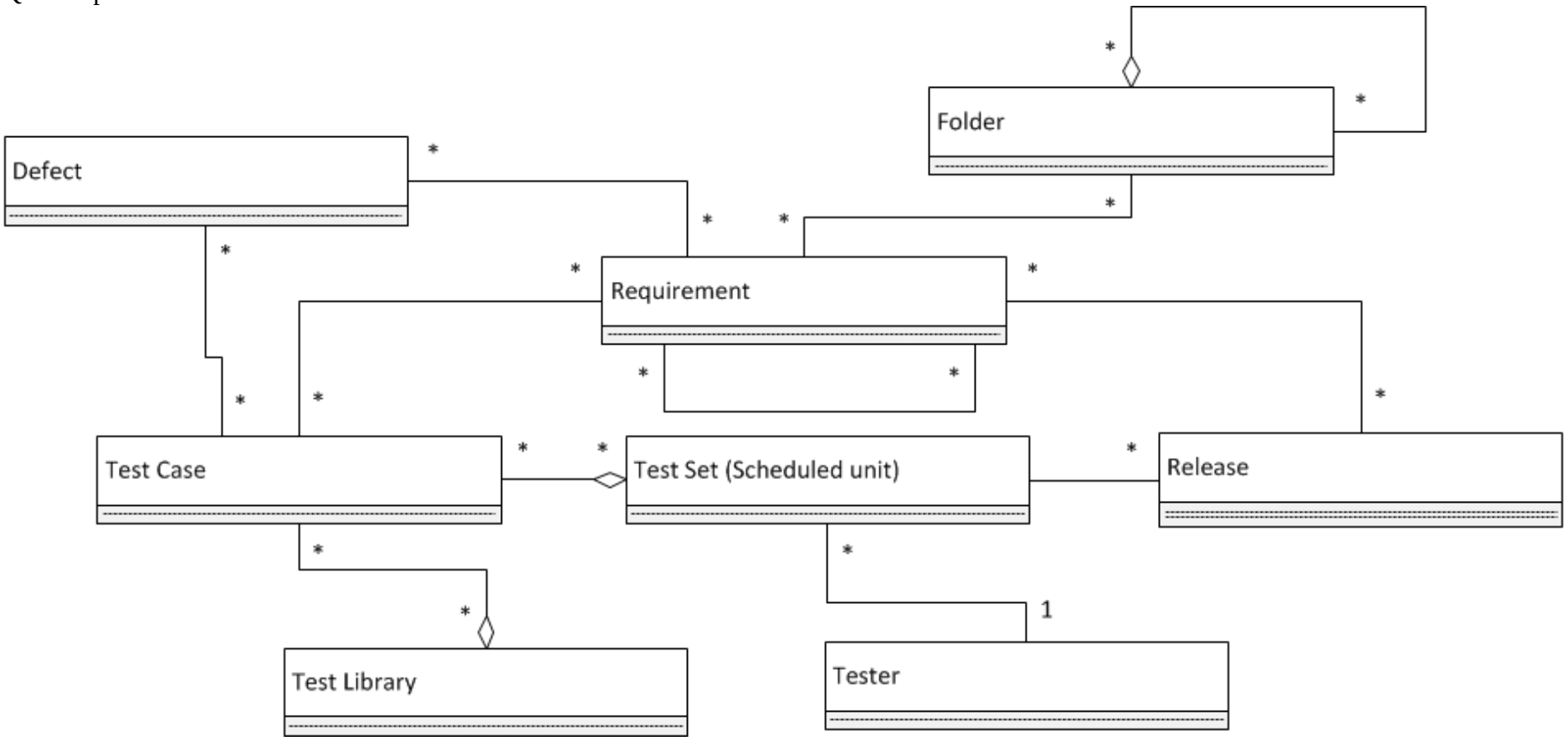
Appendix F: Structure visualizations
SilkCentral:



Zephyr:


Folder
Requirement
Test Case
Phase (scheduled unit)
Release
Defect
Cycle
Tester
*
1

QAComplete:


Folder
Defect
Requirement
Test Case
Test Set (Scheduled unit)
Release
Test Library
Tester
1