Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Heikki Honkanen

Investigating Effects of Test Automation In a Large Software Project: A Case Study

Master's Thesis
Espoo, March 3rd, 2016

Supervisors: Professor Marjo Kauppinen
Advisor: Marko Klemetti M.Sc. (Tech.)

Aalto University
School of Science

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author:	Heikki Honkanen		
Title:	Investigating Effects of Test Automation In a Large Software Project: A Case Study		
Date:	March 3rd, 2016	**Pages:**	72
Major:	Software Business And Engineering	**Code:**	T3003
Supervisors:	Professor Marjo Kauppinen		
Advisor:	Marko Klemetti M.Sc. (Tech.)		

Test automation is currently a very popular method used in software development. The effects and effectiveness of test automation is a widely discussed topic, but currently little research exists on the topic. The goal of this thesis was to investigate effects of test automation in a large software project. The research was carried out as a case study. It included a literature review and an empirical study which consisted of data analysis and qualitative research in the form of interviews.

The main benefits of test automation found in this research were: faster execution, better attention span of testers, consistency in test execution, regression tests, communication and better use of resources. Test automation also brought intangible benefits, such as increased motivation to testers and ability to run tests that are not feasible to run manually, for example creating thousands of users. The main pitfalls of test automation found were: increased initial effort in testing, more test maintenance, difficulties with volatile environments in testing, less new bugs found and inability to replace human testers. Most benefits and pitfalls were also present in the results of the literature review, but some of them were hardly mentioned.

According to the literature review, test automation effectiveness can be measured by return on investment. The results from the data analysis show a positive return of investment for the case project. Additionally, if test automation brings monetary benefit, it is usually realized in the long run. In the research case example during 18 months.

The conclusion of this thesis is that that test automation can offer tangible and intangible benefits when implemented over manual testing, usually in the longer run. Intangible benefits can be difficult to quantify in terms of money, but should not be disregarded when deciding whether to automate or not.

Keywords:	test automation, test automation effects, return on investment, testing
Language:	English

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan koulutusohjelma

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Heikki Honkanen

Työn nimi:
Testiautomaation vaikutukset: tapaustutkimus suuressa ohjelmistoprojektissa

Päiväys:	3. maaliskuuta 2016	**Sivumäärä:**	72
Pääaine:	Ohjelmistotuotanto- ja liiketoiminta	**Koodi:**	T3003
Valvojat:	Professori Marjo Kauppinen		
Ohjaaja:	Diplomi-insinööri Marko Klemetti		

Testiautomaatio on nykyään laajalti käytetty menetelmä ohjelmistokehityksessä. Testiautomaation vaikutuksista ja tehokkuudesta on käyty laajaa keskustelua, mutta siitä löytyy rajallisesti tutkimusta. Tämän diplomityön tarkoituksena oli tutkia testiautomaation vaikutuksia suuressa ohjelmistoprojektissa. Työ toteutettiin tapaustutkimuksena, johon kuului kirjallisuuskatsaus ja empiirinen osio, jossa toteutettiin data-analyysiä ja haastatteluita.

Tässä työssä testiautomaation pääasiallisiksi hyödyiksi havaittiin nopeampi testien ajo, testaajien parempi keskittymiskyky, testien ajon yhdenmukaisuus, regressiotestaus, kommunikaation paraneminen ja resurssien parempi käyttö. Testiautomaatio näyttää tuovan myös aineettomia hyötyjä, kuten testaajien parempi motivaatio ja kyky ajaa testejä joita on epäkannattavaa tai mahdotonta tehdä käsin, esimerkiksi tuhansien käyttäjien luominen. Testiautomaation pääasiallisia haittoja joita tässä työssä löydettiin olivat testauksen kallis aloittaminen, enemmän testien korjaustyötä, ongelmat epävakaiden järjestelmien kanssa, vähemmän uusien ongelmien löytämistä ja testiautomaation kyvyttömyys korvata manuaalisia testaajia. Kirjallisuuskatsauksen perusteella monet hyödyistä ja haitoista ovat relevantteja, kun taas osa mainittiin vain pinnallisesti.

Kirjallisuuskatsauksen perusteella testiautomaation tehokkuutta voi mitata sijoitetun pääoman tuotolla. Data-analyysin tulokset viittaavat positiiviseen tuottoon tapaustutkimuksen kohdeyrityksessä. Lisäksi, jos testiautomaatio tuo rahallista hyötyä, se yleensä toteutuu pidemmällä aikajänteellä. Kohdeyrityksessä 18 kuukauden aikana.

Työn johtopäätös on, että testiautomaatio voi tuoda aineellisia tai aineettomia hyötyjä manuaaliseen testaukseen verrattuna mutta usein pitkällä aikavälillä. Aineettomien hyötyjen arvioiminen rahassa on vaikeaa, mutta niitä ei kannata jättää huomiomatta automaatioprojektiin ryhtyessä.

Asiasanat:	testiautomaatio, testiautomaation vaikutukset, testaus
Kieli:	Englanti

Acknowledgements

I would like to thank my supervisor Marjo Kauppinen for her valuable support during the process of writing this thesis. I would also like to thank my colleagues and instructor at work for good tips and support in the writing process. Many thanks to everyone who took the time to read and comment my work during the research process!

Finally, the biggest thanks I'd like to give to Ida for her tireless efforts in pushing me to do the work and for her countless comments, corrections and suggestions to the content of this thesis.

Helsinki, March 3rd, 2016

Heikki Honkanen

Contents

Chapter 1

Introduction

1.1 Motivation

Software testing has always been an important part of software development, both as a development and a business tool. Traditionally, testing has been done manually, which is slow and costly. However, nowadays the schedules and budgets of software projects are being cut back, and agility in implementing software is becoming more and more important (Dustin et al., 2009). This leaves less and less room for development, which usually is cut back in testing efforts (Dustin et al., 2009).

Test automation is an alternative manual testing, and in literature its theory exists from the mid and late 90s. Most literature focuses on implementation of automation, but few take its effects in software projects into account. Test automation has been suggested as the salvation to some of the problems of manual testing: it promises more testing executed faster (Hayduk, 2009; Ramler and Wolfmaier, 2006). On the other hand, automated testing is more difficult to create (Dustin et al., 2009; Hoffman, 1999; Ramler and Wolfmaier, 2006) and maintain (Fewster and Graham, 1999; Hoffman, 1999; Bach, 2003), and requires upfront investments (Ramler and Wolfmaier, 2006), which induces reluctance in using and implementing automated testing. The question often is: will test automation save enough time and money for it to be a feasible option in reducing costs and/or shortening the project schedule?

This question had been asked in literature, and there are few studies on the effectiveness of test automation (Fewster and Graham, 1999). However, the results of these studies are often vague, and in most cases offer no em-

pirical study to back their research. The motivation of this thesis is twofold: to study literature for ways of examining the effectiveness of test automation and to provide empirical proof to either its benefits or disadvantages.

1.2 Research problem and questions

The research problem of this thesis is:

> How does the implementation of test automation affect software projects?

The research questions that are asked in order to answer the research problem are:

- RQ1: How can test automation effectiveness be measured?
- RQ2: What is the impact of test automation to software project profitability?
- RQ3: What are the benefits and pitfalls of automated testing?

1.3 Subject focus

This thesis focuses on the effects of test automation. It does not pay attention to:

- How should test automation be implemented for it to be successful?
- What are the reasons that usually make test automation fail?

1.4 Thesis structure

Table 1.1 summarizes which chapters of the thesis answer to each research question.

	Literature	**Empirical**
RQ1	3.3	
RQ2		4.1.1
RQ3	3.2.4, 3.2.5	4.1.2

Table 1.1: Thesis structure.

Chapter 2

Method

This section describes the research methods used in this thesis. The thesis consists of a literature review on software automation, and a case study, in which theories from the literature review were applied to a test automation case. Additionally, interviews were conducted to people involved in the case study project. Image 2.1 shows the basic outline of the research and how the different parts relate to each other.

The research process began by studying the research methods of literature and case studies. Having considered the research approaches, a thorough literature review was made, including basic theory of testing (only in a minimal scale), test automation and automation effectiveness. The primary results of the literature review were collected into a list in order to begin the empirical part of the research.

The empirical part consisted of making return of investment calculations on the case company and their automation project. The results of the literature review, mainly automation theory and possible means of calculating its effectiveness, were used in this part. After completing the return on investment calculations, a study on interview methods was made in order to gain knowledge on how to conduct reliable interviews (covered later in this chapter). The results from the interview method study were used to plan and execute the interviews.

The research method selected for this study was a case study, which is a strategy for doing research involving an empirical investigation of a real life phenomenon (Saunders et al., 2011). The case study method was also appropriate since it allows multiple data collection techniques that can be used in combination (mixed methods), such as interviews and data analysis,

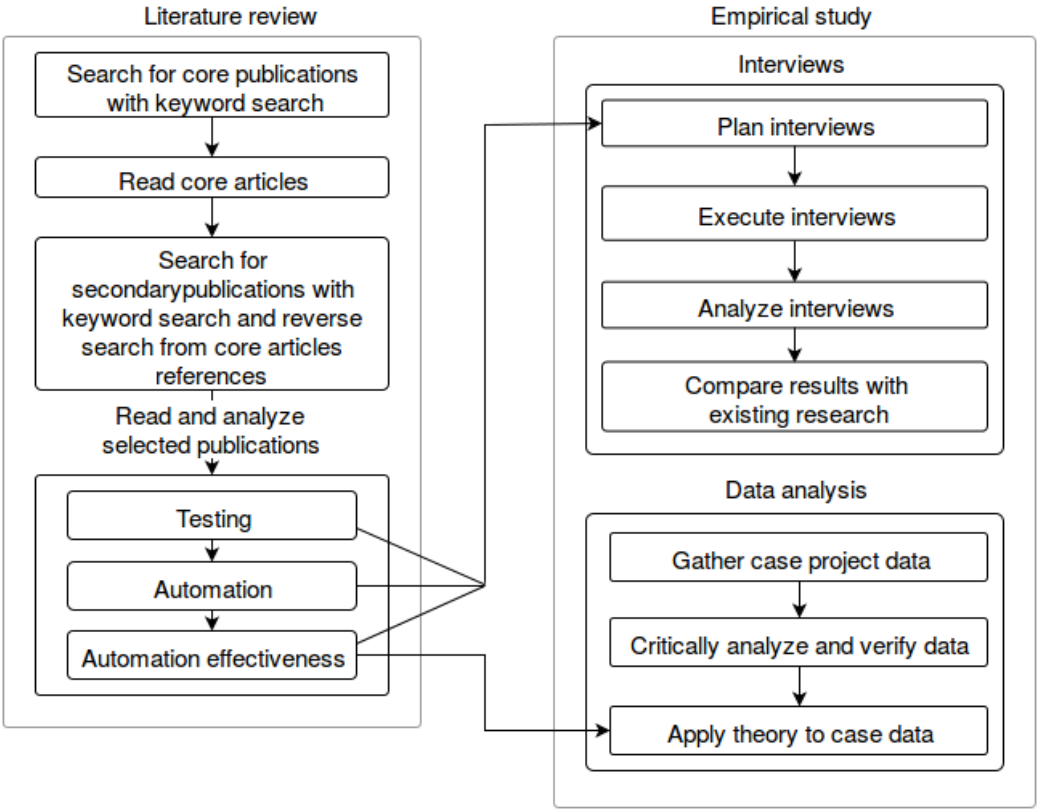

Figure 2.1: Research Process

which were used in this research (Saunders et al., 2011). Triangulating data is a phenomenon in case studies, which was also used in this thesis, where connections are made between different sources of data (Saunders et al., 2011). For example, collecting data in an interview to strengthen the data collected by analysing business data.

2.1 Research Approach

The four main research philosophies described by (Saunders et al., 2011) are positivism, realism, interpretivism and pragmatism. **Positivists** think that only phenomena that can be observed lead to credible data (Saunders et al., 2011) and that only that which can be measured can be studied (Eriksson and Kovalainen, 2008). Generating a research strategy to collect such data likely uses existing theory to form hypotheses. These hypotheses are then

either confirmed or refuted, adding into the existing body of theory and generating further research (Saunders et al., 2011). In a positivist approach the aim is to be value-free, i.e. the researcher should avoid biasing the results with their own values (Eriksson and Kovalainen, 2008; Saunders et al., 2011).

Similar to positivism, **realism** is the belief that there exists a world independent of the human mind (Eriksson and Kovalainen, 2008; Saunders et al., 2011). However, realism suggests that knowledge about the world is constructed socially (Eriksson and Kovalainen, 2008). This means that our senses show us the truth (Saunders et al., 2011) and that the world would still exists without any human mind perceiving it. **Direct realism** claims that the world is as we see it, while **critical realism** argues that we experience sensations and images of the real world, not the actual world itself (Saunders et al., 2011).

Interpretivism is based on understanding how people, as individuals or in groups, interpret and understand social settings around them (Eriksson and Kovalainen, 2008). Central in interpretivism is to adopt an emphatic stance, to enter the social world of our research subjects and to understand the world from their point of view (Saunders et al., 2011).

Finally, **pragmatism** questions if only one position must be adopted in research among epistemology, ontology and axiology - each might be better suited to answer certain types of question (Saunders et al., 2011). It emphasizes that choosing between one or the other in reality might not be possible, and that qualitative and quantitative research can be (mixed methods) can be appropriate for research (Saunders et al., 2011).

Saunders et. al (Saunders et al., 2011) describe **epistemology** as concerning what is acceptable knowledge in a field of study: it raises the question of what information is relevant to each person. Epistemology also defines criteria by which knowledge is possible (Eriksson and Kovalainen, 2008). For example, in a quantitative study, the analysis of 'real' data of a factory's process is more important than the analysis of the feelings of the workers. Also, if the data about the feelings of the workers can be quantified in a table for instance, it might have more authority in the eyes of a quantitative researcher.

Ontology is concerned with ideas about existence, the relationship between people and society and the world in general (Eriksson and Kovalainen, 2008). Ontology can be divided into **objectivism**, which views social enti-

ties as external from social actors and their activities, and **subjectivism**, in which social phenomena are constructed from perceptions and actions of social actors (Eriksson and Kovalainen, 2008; Saunders et al., 2011).

Axiology studies judgements about values and notes that the values of the researcher are of great importance in every stage of the research process if they with that their results are credible (Saunders et al., 2011). Even the choosing of the research topic, the research philosophy and the data collection method present a choice, which then reflects the researcher's values (e.g. some questions are more important than others) (Saunders et al., 2011).

The philosophy in this research was both interpretivistic and positivistic. In the interviews (qualitative study) it was necessary to be able to understand the case in the interviewees' point of view (interpretivistic) and also to keep a value free and objective, non-biased stance in order to generate credible results (positivism). Additionally, the case data study (quantitative) required measuring real life events, which also leaned towards positivism. The epistemological and axiological views were also be adopted in order to evaluate the results and the data of the interviews and the case study.

2.2 Literature Review

The literature used as the background study for this thesis consisted primarily of academic research papers, books and web references. The initial articles and books were searched from online databases of scientific journals and papers (IEEE Xplore, Emerald Insight, and Google Scholar) using keyword search. Samples of main keywords are listed in table 2.1. The main keywords were used for search as is and one or more complementing keywords were used in many combinations with the main keywords. After an initial set of articles and books were found, they were briefly analysed according to the content of their abstract, introduction and results sections and with in text keyword search. Articles and books from this search were selected as the core material (Hoffman, 1999; Fewster and Graham, 1999; Bach, 2003; Dustin et al., 2009; Kelly, 2004) of the literature review.

From the set of core material, other publications were found by reverse-searching from their references lists. As it was soon quite obvious that a large proportion of literature on test automation was in books, further search of new material was also directed to databases of book releases, such as Safari Books, Sage Publications and Addison Wesley. From the first thorough read

Main keywords	Complementing keywords
test automation, software test automation, automated testing, testiautomaatio	benefits, pitfalls, ROI, methods, business, effectiveness, techniques

Table 2.1: Summary of main keywords in publications search.

of core material their references lists were read carefully in order to find more references. After reading the core material, the references from them were analysed in the same way as the core material was.

The validity of the material was evaluated by critically analysing the publisher, the origin (web page, publications database, and other media), content of the text and the amount of references made to the material in other publications. Publications in popular databases and web sites generally approved by the community (such as IEEE) were usually approved as is. Other publications available in online book stores or other web pages were accepted if they were referred to in other selected articles, in addition to the other mentioned criteria.

2.3 Empirical Research

The empirical section of this thesis consisted of analysing a large test automation project conducted for a telecommunications company in Finland. In the empirical study, the theories studied in the literature review were applied into the project's context. The empirical study also consisted of interviews conducted on the individuals who implemented the test automation project.

2.3.1 Case Description

The case study was based on an automation project being conducted for a large telecommunications company (henceforth referred to as the *customer*) in Finland. The customer was developing a CRM platform, which contained apprixomately 500 manual regression tests at the time of the start of the automation project (summer 2014). None of the tests were automated. As the development of the platform continued, the amount of regression tests

was constantly growing and testing started to become a bottleneck in the release cycle. Scaling by adding more testers did not bring the additional capacity needed and something else needed to be done.

As a solution, a project of automating the regression tests was proposed by the implementing company. The proposal was to implement an automation infrastructure, including a continuous integration server and a reporting view, and to reduce the amount of manual testing effort by automating regression tests. The target was to reduce about a one week's worth of manual testing per cycle by the end of the year 2015. By the time of writing this thesis testing was done in 10 separate systems and about 10% of the manual regression tests have been automated.

2.3.2 Project Data Collection And Analysis

The data collected from the project was based on the data-archives of the implementing company and the notes and experiences of the developers who implemented the automation project. All sensitive financial data has been altered to use an industry standard (for example hourly billing) in order to protect the privacy and business of the customer and the provider of the project.

Although many figures in the data (such as work time allocations) were based on an estimate of the developers, the total amount of money spent on the project is known accurately. Thus, all data is based on actual financial records, but the ratio in which they are divided into each activity is a professional estimate.

The case study data analysis was carried out by listing all factors (test implementing costs, test execution costs, design etc.) included in the project costs and benefits and then analysing them with the theory provided by the literature review. After the factors had been identified and quantified, calculations based on the literature review were made on the data.

2.3.3 Interviews

The purpose for conducting interviews in this research was to gain knowledge on the benefits, pitfalls and costs of test automation from the professionals implementing the automation project. The interviews were conducted on 5 consultants who implemented the automation project. The interviewees were selected on the gounrd that they were the team who implemented the

automation project studied in this research and had the best inside knowledge on the project itself. The interview questions can be found in appendix 1 (chapter 7.1).

The interviewees were all between 25 and 35-year-old males, who work as test automation consultants. Two of the interviewees were senior consultants, with approximately 7 years experience with test automation. The rest were consultants with 1-3 years of experience. All the interviewees had worked with implementing the automation project (writing automated tests), and three of the consultants had also worked in designing the automation project, the tests and in maintaining the automation system.

Interviews are one of the most commonly used tools of gathering information in a multitude of disciplines (Ruusuvuori and Tiittula, 2005). In this thesis interviews were conducted in order to gain a qualitative perspective in the data collection. According to Corbetta (Corbetta, 2003) qualitative research is a method to gain insight of the subject's perspective of the question or object under research. The interviewee is free to express their thoughts and experiences. Figure 2.2 presents Corbetta's categorization of interview types.

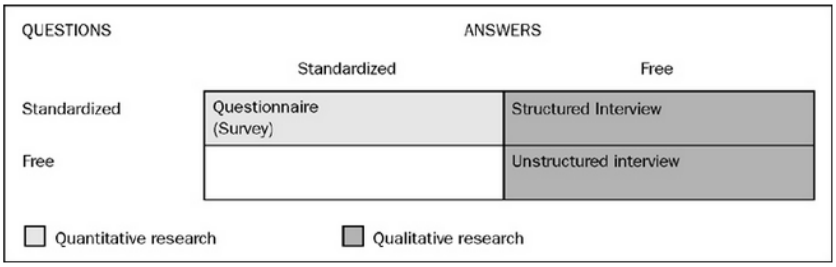

Figure 2.2: Techniques of data collection through questioning (Corbetta, 2003)

Interviews are generally categorized in three groups: structured (or standardized) interviews, semi-structured (or general) interviews and unstructured (or informal) interviews (Corbetta, 2003; Ruusuvuori and Tiittula, 2005; Turner III, 2010). In structured interviews all respondents are asked the same questions in the exact same way and sequence (Corbetta, 2003). However, the questions are formulated so that the respondents can answer freely, making them so called open-ended questions (Turner III, 2010). This means that while the questions have a strict form and are standardized, the

answers can still vary depending on the respondent. It also means that the setting and form of the questions must remain as unbiased as possible to elicit spontaneous and consistent answers from all respondents (Ruusuvuori and Tiittula, 2005). Due to the rigid structure, structured interviews can be an attempt to mediate between qualitative and quantitative research (Corbetta, 2003).

The semi-structured interviews have a less rigid structure, and the interviewer ensures that general topics are discussed with each respondent by asking questions (Ruusuvuori and Tiittula, 2005; Turner III, 2010). The semi-structured interview usually has a basic outline and the interviewer is free to develop new themes during the interview if necessary (Corbetta, 2003). The benefit of this type of interview is the flexibility (Turner III, 2010) and the possibility to ask (or probe) for more answers with extra questions (Corbetta, 2003) more than in the structured interview, where extra questions are also outlined. A drawback of semi-stucured interviews is that since the wording, order and posing of the questions is not standardized, the answers to each question cannot be assumed to be consistent (Turner III, 2010).

Unstructured interviews are close to a general discussion, but in a decided topic, in which no specific questions are asked (Corbetta, 2003; Ruusuvuori and Tiittula, 2005), but are formulated as the conversation moves forward and the interview process relies on the interaction between the interviewer and the respondent (Turner III, 2010). The interviewer is thus mostly responsible for keeping the discussion on the right track, if the respondent strays off topic (Corbetta, 2003). The lack of structure can be both beneficial, as it allows a very free conversation, but also unreliable since the questions are not asked consistently making it difficult to quantify the data (Turner III, 2010).

For the purpose of the interviews in this research, the semi-structured interview appeared to be the best option since the target was to gain insight on the aspects of test automation. The semi-structured interview gives a chance of probing for more elicitation on the answers if needed. On the other hand, since the answers would need to be categorized as clearly as possible for the data to be as consistent as possible, the structured approach could also be used. However, the semi-structured interview was used since it provides more qualitative answers which are needed for the purpose of this research. The qualitative answers provided a *"human point of view"* to the statistical analysis made in this thesis complementing the quantitative results of the research by verifying them with real life experience.

Although the questions and the ordering of the questions do not need to be standardized for the interviews, the bias of the interviewer (i.e. leading the interviewee to an answer by wording or ordering of the questions) should be left out. The biasing of questions and the answers was acknowledged in the interviews and it was avoided as strictly possible in order to gain neutral data. Sometimes the questions can be asked in a way that they lead the respondent to an answer (Ruusuvuori and Tiittula, 2005). For example, asking if the respondent *if they can see a car outside the window*, will prompt for a yes or no answer compared to asking *what do they see outside* which does not set any form or limitation to the answer. This kind of forming in the interview questions is important to get as unbiased answers and thus create meaningful data from the answers.

2.3.4 Interview Data Analysis

Ruusuvuori and Tiittula (2005) raise recording and transcribing the interviews as an important part of analysing the interview data. Firstly, recording interviews is usually beneficial if it is possible, since it allows the interviewer more time and focus to examine the respondent in the interview process, and does not set bias on the respondent's answers with their own actions (Ruusuvuori and Tiittula, 2005). For example, the respondent might develop patterns in answering if the interviewer only makes notes to certain types of answers. Secondly, listening to the recording might reveal topics, answers and behaviour that was missed in the interview process, such as the respondent's hesitation, delays, elicitation of answers and possible situations in which the interviewer has guided the respondent to an answer (by accident or intentionally) (Ruusuvuori and Tiittula, 2005).

The results of the interview should also always be transcribed (meaning to write down the interview from the recording) as it can serve as a good way of learning more from the answers and might reveal more detail and elicitation to the respondent's answers (Ruusuvuori and Tiittula, 2005). From the transcription, topics and factors considering test automation will be identified and quantified as accurately as possible, and then compared to the analysis done in the literature review section.

The analysis of the interviews in this research began with transcribing the interviews word-for-word immediately after the interview if possible, to be able to write notes of the interview as comments with the transcription. After all of the interviews were transcribed, the relevant points and answers

of each interview were highlighted and when needed given a *code*, meaning a certain opinion is given a name for future reference. After all interviews had gone through this primary analysis the answers and highlighted areas of the transcriptions were collected to a table. Based on the aforementioned coding and their appearances in each interview, a collection of the primary and secondary points in all interviews and the frequency in which they appear in all interviews was created.

Chapter 3

Literature Review

This chapter provides an introduction to software testing and a deeper explanation on test automation based on a thorough literature review. The chapter begins with a look on testing in general and then moves to test automation. The test automation sections covers basics of tst automation and automation implementation, automation benefits, pitfalls and measuring automation efficiency.

3.1 Software Testing

As software systems are getting more and more important for organizations and individuals, the need for software quality is also on the rise (Laukkanen, 2006). Software defects cause great losses, and as software systems are getting more and more complex and the losses will increase in case quality does not improve with increasing complexity (Laukkanen, 2006). In literature, the amount of testing done in software projects is estimated to be 20% (Graham and Fewster, 2012), although claims are made for the effort to be 30-50% (Dustin et al., 2009) or even 50% and above (Ramler and Wolfmaier, 2006). These numbers are high, which leads to the fact that there is plenty of room for improvement in the efficiency of testing. This section covers the basics of software testing. Test automation will be introduced in a section of its own since it is the central theme of this thesis, and thus requires a more thorough introduction than the basics described in this section.

3.1.1 Definition

Software testing is defined in the IEEE standard Software Engineering Body of Knowledge (Bourque and Fairley, 2014) as:

> Software testing consists of the dynamic verification that a program provides expected behaviours in a finite set of test cases, suitably selected from the usually infinite execution domain.

The word dynamic means that the state of the program is assumed to be the same with the same selected inputs of the test, since some programs might react to the same inputs differently if the state of the program changes. Nowadays software testing has become more about detecting errors rather than fixing them by planning and implementing testing throughout software development, not just after coding (Bourque and Fairley, 2014). This approach of error prevention is an approach to software quality, which relates to another definition of software testing by Myers et al. (2011), who see the objective of software testing as adding value to the software rather than trying to prove that the software works correctly. This means that testing should be done to with the assumption that all software has errors, and testing is the way of finding them and should be done in an effort to find errors (Myers et al., 2011).

Myers et al. (2011) offer an explanation to this definition as the result of goal-oriented psychology of humans, which if given the task to prove that something works, it is tested to that end, not to find possible errors. This kind of definition offers a more value adding perspective to software testing since it has the business quality component as a driver for testing, rather than the "let's just make sure it works" -attitude that would easily let defects slip by. While from the customer's perspective, the first definition to testing is more relevant – "It is what we pay for, a functioning piece of software". The latter definition is more to the liking of software experts, since by finding and fixing as many defects from the software, the desired state of the first definition will be accomplished. However, this does not work the other way around. This association is illustrated in figure 3.1. Also, from the professional point of view, the software created should not only work as expected when used correctly, but not to work in unexpected way when used incorrectly, which cannot be ensured by the "testing that it works" -ideology.

3.1.2 Types of Testing

Testing is usually divided into two types of testing: black-box and white-box testing. The difference between these types is summarized in figure 3.2. **Black-box testing** (also known as data-driven or input/output-driven testing) means that the internal structure or logic of the system under test is not known to the tester. In this type of testing the concern is to verify that

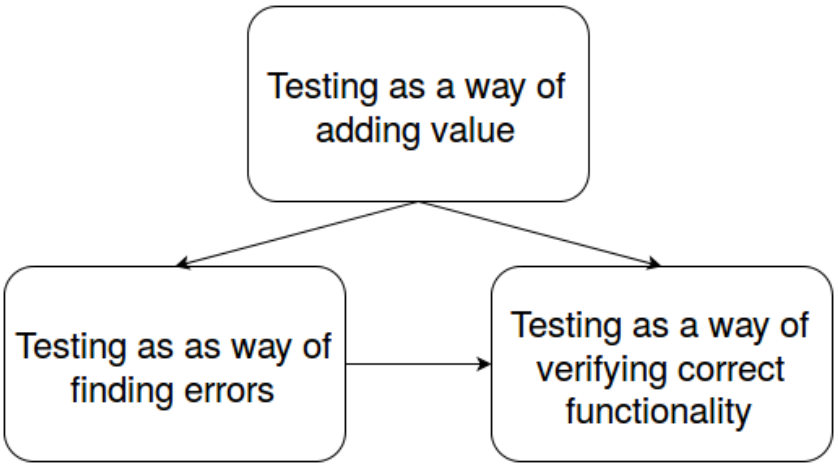

Figure 3.1: The relation of testing philosophies as explained by Myers et al. (2011)

the system works according to its specification, from which the test data for the tests is derived. The main problem in black-box testing is that to exhaustively test the system, the tester would need to run it with every possible combination of valid inputs. This kind of testing quickly escalates to a virtually infinite amount of test cases. The main focus in black-box testing should thus be in maximizing the value generated by the tests compared to the cost of testing. (Myers et al., 2011)

White-box testing (or logic-driven testing) allows the examination of the program code (the internal structure) and the test data is drawn from the program's logic. Exhaustive testing in the white-box situation seems simpler, since it will be easier to test all statements atleast once. However, this is not enough, since statements might have multiple consequences in the program execution. For this reason, every *logical path* in the code should be executed once, which again leads to a very high number of test cases, and an impractical solution in large software systems. (Myers et al., 2011)

3.1.3 Levels of Testing

Levels of testing can be divided by the *target* of the test level (a module, a group of modules or the entire system) or the *objective* (the system's objectives) of the test (Bourque and Fairley, 2014). The separation by the *target* of the test level means that the test types are separated according to the different types of targets in the system (illustrated in figure 3.3). These can

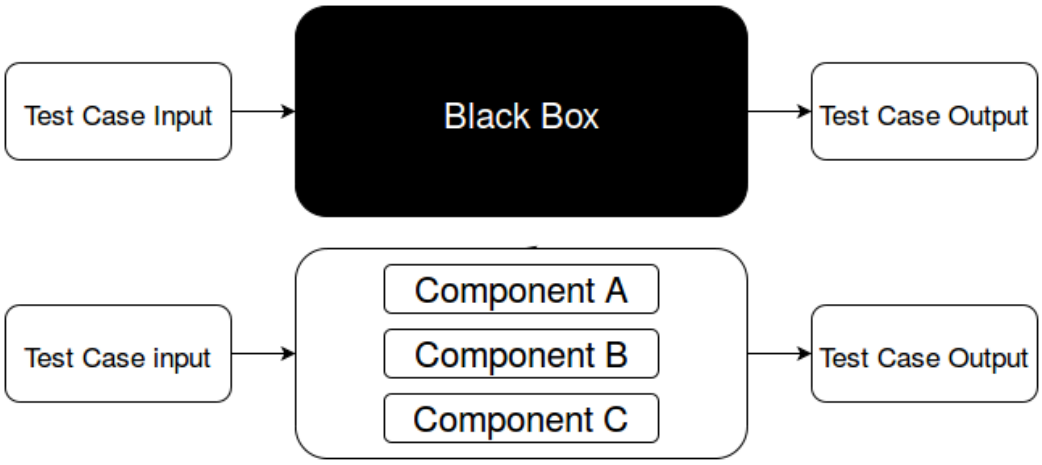

Figure 3.2: Types Of Testing

be functions (or methods), components, classes, interfaces or even the entire system. Three main types for this kind of testing are unit testing, integration testing, system testing (Bourque and Fairley, 2014) and acceptance testing (IEE, 1990).

Unit testing concerns testing the individual units of the software, such as functions or modules, to see that they meet the specification, usually done by the developers (Bourque and Fairley, 2014). Integration testing verifies the correct functioning of the interfaces between different components of the system(Bourque and Fairley, 2014). While unit and integration testing usually detect most defects in the system, system testing is concerned with the non-functional aspects of the software. These aspects can be security, performance, usability and other possible non-functional requirements (Bourque and Fairley, 2014). In acceptance testing the testing of the software is done to determine whether or not the system meets its requirements and whether or not the customer can accept the system or a component (IEE, 1990).

The *objective* of the test level refers to testing different properties of the software, such as performance, functional, reliability, usability and security. These objectives often change with the test target, which means that different objectives are tested at different levels of testing.

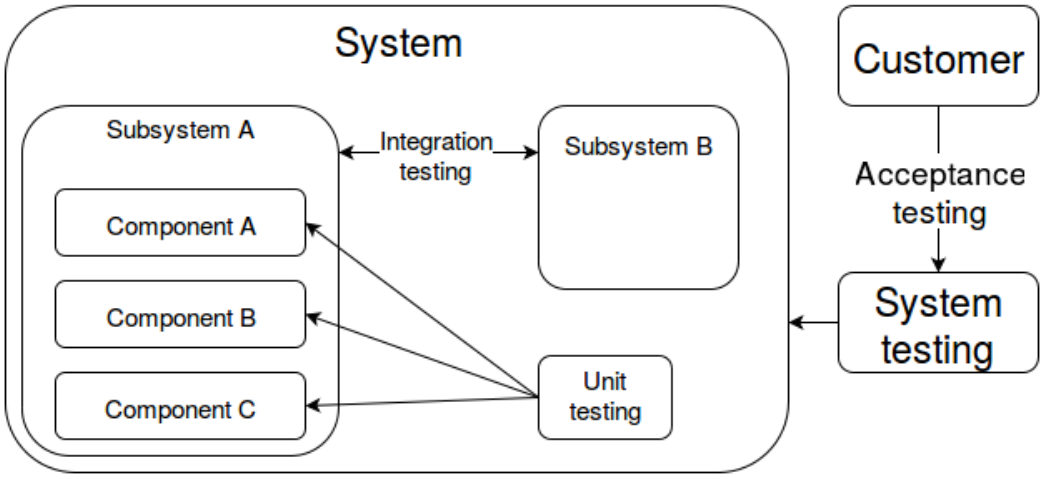

Figure 3.3: Levels Of Testing

3.1.4 Manual Testing

Manual testing is software testing where the tester has to put theirself in the end user's position and test the software in the way the end user would. This means that the testing process is human-present and that the tester uses real data for input and real environments to test the software to create actual usage scenarios. Manual testing is often used to find bugs in the business logic of the software, i.e. the code that implements the user requirements. (Whittaker, 2009)

Manual testing is required when there are just too many possible scenarios for automation, for example large ranges of input with many fields of input, so the tester has to use their wits to sort out the most possible ranges of inputs. Often, manual testing is guided by a script or a test document that gives instructions and details on how to run the tests. These documents can be very specific, giving step-by-step instructions on input, actions and expected outputs, or they can be vague, giving an instruction on what to do, not how to do it. (Whittaker, 2009)

In manual testing three main levels of testing are recognized: ad hoc (or exploratory) testing (no script), vague scripts and detailed scripts (Fewster and Graham, 1999). In ad hoc testing testing is done without any plan or guide. Instead, the tester will have to think up different scenarios and inputs and to check that the program works as specified by requirements (Fewster

and Graham, 1999). This kind of testing can be particularly effective since it utilizes the creativity of the tester (Itkonen et al., 2007) and is best suited for testing where the interfaces change often, such as modern web testing (Whittaker, 2009).

Vague scripts usually offer a low level of instruction on what to do but not how to do it, e.g. "try incorrect inputs", but the inputs are not given and still have to be made up by the tester. Detailed scripts offer both the *what* and the *how*: what to test, the inputs to test with and the expected results. Detailed scripts are the most boring ones for testers, as they leave no room for imagination but also the easiest ones to start an automation process from, since all elements of the tests are specified. (Fewster and Graham, 1999)

Manual testing often receives criticism due to being slow, ad hoc and non-repeatable (Whittaker, 2009). For this reason, it should be used to in conjunction with automated testing and to let automation do as much of the repeatable work as possible (Rantanen, 2010). However, manual testsare considered better in finding new bugs since bugs are most often found when the test is run for the first time (Fewster and Graham, 1999).

3.2 Test Automation

This section offers an introduction to test automation and to its benefits, pitfalls and on how to measure its efficiency. This section also briefly covers implementation of test automation and problems of manual testing. Problems of manual testing are used as a preface to test automation benefits and pitfalls.

3.2.1 Definition

In literature there exists countless definitions to test automation, but perhaps the most fitting one is by Koirala and Sheikh (Koirala and Sheikh, 2009):

> Automation is the integration of testing tools into the test environment in such a manner that the test execution, logging, and comparison of results are done with little human intervention.

From this definition we can see that test automation is not just about automating test execution, but also automating as much of the support and side activities of testing as possible even though most definitions and the common

understanding of test automation is related to test execution. Other things (besides execution) that can be automated include test data generation (creating combinations of inputs, populating databases with test data), system configuration (preserve or recreate environments), simulation (mocking system features that are not yet available, mocking network, database access etc.), results analysis, recording activities and coverage and communicating test results (Bach, 2003).

Automated testing can cover all types of testing (functional, performance, concurrency, stress etc.), all testing phases, is process-independent and generally, all tests that are currently run or can be run manually can be automated (Dustin et al., 2009). Automated testing differs from manual testing in, for example, the following ways (Dustin et al., 2009):

- Enhances manual testing by automating tests that are difficult of impossible to do manually.
- Is in itself software development.
- Does not replace the need for manual testers' analytical skills, test strategy know-how, and understanding of testing techniques.
- Can't be clearly separated from manual testing. Instead, both automated testing and manual testing are intertwined and complement each other.

Many manual tests can be converted to automated tests but they often need to be adjusted to accommodate automation (Dustin et al., 2009). The objective of automated testing by Dustin et. al (Dustin et al., 2009):

> The overall objective of AST is to design, develop, and deliver an automated test and retest capability that increases testing efficiencies; if implemented successfully, it can result in a substantial reduction in the cost, time, and resources associated with traditional test and evaluation methods and processes for software-intensive systems.

3.2.2 Implementing Automated Testing

Dustin et. al. (Dustin et al., 2009) present six keys for successful automation:

- "Know your requirements"

- Develop an automation strategy
- Verify the automation framework
- Track progress and adjust accordingly
- Implement automation processes
- Appropriate usage of skills

The first key is to know the requirements of the system under test. The requirements of the system under tests serve as the baseline for the whole automation process, and thus is one of the most important factors that will impact the success of the automation project (Dustin et al., 2009). For this reason all requirements should be documented in a clear and concise manner so that they are accurate (Dustin et al., 2009).

The next step is to develop an automation strategy. The automation strategy can be seen as a blueprint, which defines the scope, objectives, approach, framework, tools, environments, schedule and personnel requirements of the automation effort (Dustin et al., 2009). It should also include information on how the tests are to be maintained, and estimates of costs and benefits of the project (Zallar, 2000). In practice, the strategy should start out with small steps, meaning that small areas should be tested and verified their effort and payback instead of trying to automate everything at once (Zallar, 2000). If the automation strategy is not clear before implementation, the result can be a a large amount of test code that no one understands (Bach, 1999).

Third is the verification of the automation framework. The framework selected to the automation project should be verified so that is works as expected and allows for modifications or extensions, which are commonly required even if a ready-made framework is used (Dustin et al., 2009). Tool selection should be done carefully, and trial versions used before making the decision to buy (Bach, 1999). Bach (Bach, 1999) lists guidelines in tool selection: Does the tool have all features needed? Is it reliable? Does it work beyond examples? Is it easy to learn and operate? Is it powerful enough? Does it simulate actual users well?

Tracking progress and making adjustments means that no matter how well something is prepared and planned, something will always go wrong (Dustin et al., 2009). When the testing process is monitored effectively, adjustments, such as fixing bugs, adjusting schedules and reducing goals, can be

made accordingly to avoid problems in the future (Dustin et al., 2009). While tracking and prevention are important, still things can go wrong. Lessons learned should be collected, for example by using root cause analysis to find the root of the problem and either eliminate it or learn to mitigate it (Dustin et al., 2009).

The fifth key is to implement an automation process. The process itself should be lightweight, well defined and structured but contain as little overhead as possible (Dustin et al., 2009). The whole automation process should be treated as any software development effort, which means it should include defining *what* to automate, design and execution (Zallar, 2000). The process should also be very flexible to allow for easy feedback loops (Dustin et al., 2009) and convenient to review (Bach, 1999). Also, a clear distinction between the automation process and the process it automates should be made (Bach, 1999).

Finally, appropriate use of skills means that people are at the core of successful automation implementation, and that they should be assigned according to their skills and capabilities (Dustin et al., 2009). People with various skills are needed who also understand their responsibilities and roles, which makes hiring competent and qualified people an important activity in automation projects (Dustin et al., 2009). Zallar (Zallar, 2000) also suggests that the project should have a senior automation expert, or a "champion" who has skills in project management and development, and is responsible for being managing the project and communicating between the testers (i.e. automation developers) and the developers who develop the software under test.

3.2.3 Problems With Manual Testing

Starting with *why* should test automation be introduced, the question is what problems of manual testing it can solve. Some common problems of test automation are listed in table 3.1. **Boring, repetitive and mundane work** can be a problem for many testers or developers. By manually testing the same thing over and over again, as similarly as possible, can cause tester fatigue, which is the habit of seeing the working solution of the system under test without considering the non-working paths (Dustin et al., 2009). Lack of motivation on the other hand can effect performance and accuracy and over all job satisfaction.

Non-repeatability and **non-reproducibility** mean that the tests are

Problem	Summary	Author(s)
Incentive	Manual testing can be repetitive, boring and mundane, reducing motivation and incentive to work.	Dustin et al. (2009), Malaiya (2011)
Repeatability	There is no guarantee that manual tests are run in the same sequence time they are run.	Fewster and Graham (1999), Whittaker (2009)
Reproducibility	There is no guarantee that manual tests are run with the same input every time.	Fewster and Graham (1999), Whittaker (2009)
Data-intensive tests	When tests require entering or generations lots of data, testing becomes very difficult	Dustin et al. (2009), Fewster and Graham (1999)
Slow	Testing manually is slow since everything is done by hand	Whittaker (2009)
Difficulty	Manual testing requires experience in techniques and methods in order to be effective, which are hard to learn.	Whittaker (2009)

Table 3.1: Problems of manual testing.

not always run in the same way when done manually. The first implies, that if a test fails, checking that it works after it has supposedly been fixed cannot be done with complete accuracy (unless there exists extensive test documentation of the test), since doing the ***same exact steps*** depends on the memory of the tester (Dustin et al., 2009). The latter means, that running tests will not always be done with the ***same inputs*** as in previous runs (Fewster and Graham, 1999). While these two might seem to be the same thing, consider that when a test is *repeated* correctly, the *inputs* might still be different, i.e. the process vs. the data.

Data-intensive testing can be difficult or even impossible with manual testing, such as input or performance testing (Fewster and Graham, 1999). For example, consider testing a web application with the data of hundreds

of users. This is surely slow, and might even be impossible to repeat often, atleast in practical terms of project schedule. At the very least, entering the same large inputs, or populating a test database over and over again manually will be boring and mundane, which will decrease work performance.

Manual testing is **slow** (Whittaker, 2009), since all test cases must be entered and evaluated manually. For example entering user inputs on an application takes as long as the tester types them in, rather than fractions of a second when filled automatically by a computer.

The first of the last two in the list means that manual testing is **not an easy task**. As Fewster & Graham (Fewster and Graham, 1999) describe in the types of manual testing, most manual testing requires the tester to be creative in, for example, figuring out what parameters to use when trying to break the system. This might sound like an easy task, but learning how to effectively test software in a destructive way takes experience, whereas with automation, the inputs can be randomized and the test run dozens of times.

3.2.4 Automated Testing Benefits

With the problems of manual testing considered, in literature there are numerous examples of how test automation is supposed to improve over manual testing. Some of the factors are listed in table 3.2. One of the largest benefits of test automation is that it **reduces the time spent on executing tests**. Since automated tests run a lot faster than manual tests, they run in less time (Dustin et al., 2009; Fewster and Graham, 1999; Hayduk, 2009; Adams, 2002). Shortening the test cycle allows testers them to focus on more important tasks than simply repeating tests execution (Dustin et al., 2009), such as improving test design (Fewster and Graham, 1999), doing more complicated tasks (Dustin et al., 2009) or focusing on other project (Adams, 2002). Quite obviously, this time can also be used to automate tests that have not yet been automated. Additionally, running tests faster reduces the overall time of the program development, which may allow for a faster time-to-market (Fewster and Graham, 1999). Also, running tests more often increases confidence in the program (Fewster and Graham, 1999).

Repeating the same task over and over again can be very boring an laborious, causing reduction of **attention**. Test automation offers relief in the repetitive nature of testing by removing the need to execute the same test multiple times, always in the same way (Fewster and Graham, 1999;

Benefit	Summary	Author(s)
Faster	Automated tests run much faster than manual tests when human interaction is removed.	Adams (2002), Dustin et al. (2009), Fewster and Graham (1999), Hayduk (2009) (Karhu et al., 2009)
Attention	Testers might get bored or inattentive, causing mistakes in tests, unlike automated tests.	Adams (2002), Dustin et al. (2009), Fewster and Graham (1999), Hayduk (2009)
Consistency	Automated tests are more consistent than manual tests, since machines rarely make mistakes compared to humans.	Fewster and Graham (1999), Hayduk (2009)
Capability	Tests that are impossible and time consuming for humans can be simple for machines (e.g. large input sets).	Bach (2003), Fewster and Graham (1999), Hayduk (2009)
Regression tests	Testing automatiocally when changes are made reduces the amount of bugs found later.	Dustin et al. (2009), Fewster and Graham (1999)

Table 3.2: Benefits of test automation.

Adams, 2002). The advantage in removing menial and repetitive tasks from the testers' or developers' schedules allows them to concentrate on more challenging tasks, which require more thinking and development making the work more rewarding (Adams, 2002; Hayduk, 2009). Another benefit of removing repetitive tasks from testers is that it increases accuracy of testing (Fewster and Graham, 1999) since automated tests don't get bored or inattentive (Dustin et al., 2009; **?**) and thus don't make errors based on boredom and assumptions (Hayduk, 2009). Lastly, removing menial tasks can also increase motivation of the testers (Fewster and Graham, 1999).

Automated tests are also very **consistent**. Automation is not susceptible to human error in inputs or sequences of actions (Fewster and Graham, 1999; Hayduk, 2009), which leads to more reliable test results. For example when a test has worked before, and it works now, and there is virtually no chance

that the test was executed in a different way the second time, so if the next run of the test passes, the program can be expected to work properly for the tested part.. Another benefit of test automation consistency is that the tests can be run in the same way on multiple different platforms and environments (Fewster and Graham, 1999). In this way it van be tested that the *environment* in which the program is run in has no effect on its functionality.

Test automation has the **capability** to test some features of a program that can be impossible, or at least very time consuming and difficult for human testers. Examples of these tasks are performance tests, memory leak detection tests, concurrency and entering large amounts of input or test data (Dustin et al., 2009). Automating tests that are near impossible may allow for new areas of tests to be executed, increasing confidence in the program further. Additionally, tests that require entering large amounts of input (or require large amounts of data to be created) can be difficult or even impossible to be tested manually (Fewster and Graham, 1999) (e.g. testing an online registration form with the data of at least 200 users would not be feasible). Finally, automation can cover a larger amount of input combinations, which increases test coverage (Dustin et al., 2009; Bach, 2003). Figuring out and remembering input combinations can be very difficult for human testers. For example, consider even a single function, which takes two integers as inputs. Now, if both inputs can range from 1 to 10, the amount of combinations is 100. With a third similar input, the number is already 1000.

Running **regression tests** can beneficial when developing new features. Regressions tests make sure that tests that worked on the previous version will also work on the new version, meaning that the new features did not break anything (Fewster and Graham, 1999). With automation, a full set of regression tests can be run on the changed system, removing the additional wondering of which portions of the system should be retested after the changes were made (Dustin et al., 2009). Ideally, there exists an analysis that checks, which tests are to be rerun (Dustin et al., 2009), and usually in modern regression automations this is the case by default (e.g. when a code or test file is saved, all tests related to it are run).

Malaiya (Malaiya, 2011) presents results of The Quality Assurance Institute benchmark study from 1995, involving 1750 test cases and 700 defects. The results show that while automated testing requires more work upfront, the payback is significant by the end. Results are summarized in figure 3.4. The values in the testing columns are work hours used per activity. The numbers presented by Malaiya have little explanation in the source, and the

original study cannot be found, which gives little credibility to the results. However, they point out the general direction of how automated testing is more effective than manual testing.

Test step	Manual testing	Automated testing	Percent Improvement
Test plan development	32	40	-25%
Test case development	262	117	55%
Test execution	466	23	95%
Test result analyses	117	58	50%
Defect tracking	117	23	80%
Report creation	96	16	83%
Total hours	1090	277	75%

Figure 3.4: Manual vs. Automated testing (Malaiya, 2011)

3.2.5 Pitfalls of Automated Testing

Even though manual testing has many flaws (summarized in 3.3), it still cannot be completely replaced by automated testing (Bach, 2003; Dustin et al., 2009; Fewster and Graham, 1999). Firstly, machines cannot perceive everything that humans can (Bach, 2003; Fewster and Graham, 1999). For example, humans can much easier perceive things that are very hard to "teach" to a computer (maybe even impossible), such as verify the color scheme or aesthetics of a web page layout or tests that require lots of physical action, such as testing a payment terminal (Fewster and Graham, 1999). Secondly, automation cannot replace the analytical skills, knowledge about testing strategies and techniques of manual testers (Dustin et al., 2009). This means that human testers can improvise in addition to reading instructions, for example in case of loss of network in the test situation the human tester might be able to do something about it, the machine hardly is (Dustin et al., 2009). Also, automated tests are only as good as they are created, meaning that the results of bad tests might not be worth much, which the human can analyse while executing, but the machine can't (Dustin et al., 2009).

In many cases managers and engineers underestimate the cost and required effort in implementing and maintaining automated testing (Ramler and Wolfmaier, 2006). In truth, when implementing test automation an additional level of complexity is added to testing, which increases the effort of testing (Dustin et al., 2009). Hoffman (Hoffman, 1999) presents phenomena

Pitfall	Summary	Author(s)
Cannot replace humans	Test automation is inherently incapable of some tasks and analytic that humans are capable of.	Bach (2003), Dustin et al. (2009), Fewster and Graham (1999) (**?**)
Initial effort	Implementing automation increases initial testing effort due to more complex design and implementation.	Dustin et al. (2009), Hoffman (1999), Ramler and Wolfmaier (2006) (Karhu et al., 2009)
Finding new bugs	Automated tests are not as effective in discovering new bugs as manual tests.	Fewster and Graham (1999), Ramler and Wolfmaier (2006), Whittaker (2009)
More maintenance	Automated tests require more maintenance than manual tests due to their implementation specific design.	Fewster and Graham (1999), Hoffman (1999), Bach (2003) (Karhu et al., 2009), (**?**)

Table 3.3: Pitfalls of automated testing.

concerning schedule and performance related negative impacts when implementing test automation:

- Expected *zero ramp up time*. In truth, implementing test automation automation requires selecting, installing and integrating tools and planning and implementing test cases, which often takes many times the effort of the equivalent manual tests.
- Perceived *an immediate reduction of productivity of the test organization*. This happens due to the aforementioned ramp up time of test automation, which causes a pause in testing activities.

Since the benefits of test automation can often be be actually realized in future projects (Hoffman, 1999), managers might be reluctant to accept it into a project, since it is not financially feasible in the single project's scope (Adams, 2002). The delay in automation benefits may be caused due to its initial complexity, and the time it takes to learn automation methods and tools. Yet, automation projects often increase professionalism in the test organization (Hoffman, 1999) and thus the benefits are realized better in future projects.

A false expectation of automated testing is that it will find more new bugs than manual testing (Fewster and Graham, 1999; Ramler and Wolfmaier, 2006; Whittaker, 2009), but actually a test is most likely to find a bug on the first time that it is run (Fewster and Graham, 1999). Automated tests by definition are run again and again, more as regression tests, and thus the "*first run advantage*" is somewhat lost. Also, in cases where automation is implemented on an existing set of manual tests (which often is the case (Koirala and Sheikh, 2009)), the first run advantage is lost all together. The disadvantage of automated tests is again the lack of imagination, which makes it less useful in finding new bugs (Ramler and Wolfmaier, 2006). In all testing, the amount of possible test scenarios quickly rises, and to cover them all manually or automatically can be difficult, where the brain work of the tester can help identify the most necessary scenarios (Whittaker, 2009).

Automated test cases require more maintenance (Fewster and Graham, 1999; Hoffman, 1999), especially capture/playback techniques (Hoffman, 1999), in which the actions of users are recorded and replayed (e.g. clicking a UI). This is due to the nature of automated tests, that when something is changed in the code, it might require changes to the test cases (Fewster and Graham, 1999). The reason for this is that automated tests are created to test **specific areas** of the software, with specific routines. Compared to manual testing, which usually tests that the **process** works as expected might not need to be changed, if the tested area still achieves the same goal, only does it differently. Since automation is easily susceptible to breaking during implementation, it needs to be designed to be easily changeable and robust (Bach, 2003). In some cases, automated tests may have an expected useful life of only a few test runs, while they are much more expensive to implement (Hoffman, 1999). This further increases the need for design. The question is, should the software components be designed to be interchangeable and easily testable as possible, or do the tests need to be designed to be robust.

3.3 Measuring Test Automation

The importance and reasoning for testing is often neglected by decision makers in companies since it does not provide tangible value to the project, i.e. it does not provide, for example, new functionality. The reason for this is that testing is often argumented on the basis that testing is needed for quality or confidence on the system, rather than on measures of money and business benefit. The value of testing and test automaton should for this reason be communicated in financial terms to decision makers. By having compelling

evidence in numbers that show that every penny spent on testing will yield a larger return, will most certainly guarantee more funding and schedule for testing. (Adams, 2002)

Test automation is usually an investment either in "upgrading" from manual testing to automated testing, or in the start of a project (compared to no testing). The benefit of the investment should always be calculated or estimated before making the decision whether or not to automate. When assessing the benefit of an investment, cost-benefit analysis is usually a good starting point: resources should be spent, if the expected benefits of the investment exceed the investment (Horngren et al., 2012). Cost-benefit analysis methods are useful as a guide in analysing the benefit of an investment when the required measures of benefit are quantified to a tangible form, and are often used by management and decision makers (Horngren et al., 2012).

Since the cost benefits of test automation are usually not in an easily measurable form, but are often intangible (Rantanen, 2010), the effectiveness of test automation has not been discussed in detail in literature and empirical evidence of the possible benefits of test automation are still lacking. This chapter collects from literature a number of basic measures, regarding benefit calculations of test automation to be applied in the empirical section of the thesis.

3.3.1 Return On Investment

Return on investment (RoI) is an accounting measure, where income of an investment is divided by its cost (Horngren et al., 2012).:

$$RoI = \frac{Income}{Investment}$$

RoI is the most popular method of measuring performance as it combines all elements of profitability, such as revenues, costs and investment into a single percentage number(Horngren et al., 2012). Since RoI is just a single number performance measure, it should be used in addition to other performance measures to give more accurate result s(Horngren et al., 2012).

RoI can also be expressed as the ratio of, the benefit of the investment divided by its cost (Kelly, 2004):

$$RoI = \frac{Benefit}{Investment} = \frac{\text{Cost of manual testing} - \text{Cost of automated testing}}{Investment}$$

The difference between the two are, that the first is an economic model, which only assumes that there is an monetary investment to all investments, and that the investment yields a monetary return. The latter is already applied to software test automation (or another investment of change in a process or a project), where the monetary return is replaced with the costs saved (or wasted) from implementing test automation compared to manual testing. In the case when test automation is implemented from the beginning of the project, the value of testing is used as the benefit, if automated testing is implemented during project, the trade-off compared to manual testing should be used (Hoffman, 1999).

3.3.2 Return on Investment of Test Automation

The most common method of cost-benefit analysis in literature is return on investment (RoI) (Rantanen, 2010; Hoffman, 1999; Adams, 2002). When measuring the effectiveness of test automation, there exists the problem of realistically assessing intangible benefits(Hoffman, 1999). In a management perspective, when the intangible are difficult to assess, the tangible must then take priority. Here RoI calculations become important, since they offer and easy way and effective way of identifying and explaining the effectiveness of test automation (Rantanen, 2010). Additionally, it is useful for upper management to have financial information about test automation effectiveness in order to better make the decision whether or not to introduce test automation in the first place (Rantanen, 2010). Information that can be used to assess the RoI of test automation are financial (tangible) costs and benefits, intangible costs and benefits and negligible factors, i.e. costs and benefits that are common to manual and automated testing (Hoffman, 1999).

3.3.2.1 Tangible Factors

Financial costs are typically divided into fixed and variable costs. Fixed costs include items such as expenditures for equipment, tools and training etc., namely, items that are not influenced by the amount of test runs or tests created. Variable costs on the other hand are factors that are influenced by

the amount of test runs or by the amount of tests created. (Hoffman, 1999)

There are a number of factors introduced in literature, both variable and fixed costs. Fixed costs are summarized in table 3.4 and variable costs in table 3.5.

Factor	**Definition**	**Author(s)**
Cost of hardware	Additional hardware must sometimes be purchased for test automation, since it might need more power for handling data and large amounts of test runs.	(Dustin et al., 2009; Hayduk, 2009; Hoffman, 1999; Adams, 2002)
Licences for tools and testware	Automated testing usually requires additional software and frameworks for running tests. These might not always be free.	(Hayduk, 2009; Hoffman, 1999; Ramler and Wolfmaier, 2006; Adams, 2002; Münch et al., 2012)
Tool training	The tools used by automation are not often used in manual testing, and thus may require training.	(Hayduk, 2009; Hoffman, 1999; Adams, 2002; Münch et al., 2012)
Test automation environment implementation and maintenance	The manual testing environment usually cannot be used for automated testing, or requires extensions and maintenance.	(Dustin et al., 2009; Hoffman, 1999; Münch et al., 2012)

Table 3.4: Fixed costs of test automation.

One of the most mentioned fixed cost is **new hardware** to the automation process. Since automated testing is most suitable to testing that requires repetition (Koirala and Sheikh, 2009), or handling big amounts of data (Dustin et al., 2009), this claim might still hold for example in the case of performance testing. Yet, many of the data in literature are based on technology of 5 to 15 years ago, and modern, even desktop and laptop computers are usually powerful enough to perform most kind of testing. Additionally, projects (and software companies in general at least) usually have hardware readily present, for example servers for deployment, and as testing servers,

which often have more power than personal computers. For this reason the argument for additional hardware cost might be negligible, atleast in projects that create software for a smaller user base.

Another, possibly major cost of test automation is the **testing tools**, frameworks and testware. Even though many tools are currently available, that are free (such as RobotFramework or Selenium), there are commercial applications that are used for test automation which cost money. Testware is the set of files needed for automated testing, including databases, environments and any additional software or utilities used in automated testing (Fewster and Graham, 1999). Testware can also be proprietary, such as Microsoft databases and operating systems, which need to be acquired in case the environments need to be supported.

Another possibly large cost is **training** of the test engineers. This, of course depends highly on the technologies applied, but training of "ordinary" test engineers to do test automation can take up to 6–12 months (Hayduk, 2009). Obviously many test automation disciplines can be learned by doing, and only a little introduction is necessary to get things started. Also, at the time of writing this thesis there exists many alternatives for automation and most of them have extensive tutorials and examples available.

Finally, the **implementation and maintenance** of the test environment can prove to be difficult. Manual testing relies on working with the development environment, but automated tests might require special environments to run. Especially if there is need for a new system or other new hardware to be installed. Again, modern automation alternatives are fairly flexible and often easily integrated into the development and testing environments as is, without requiring much set-up or maintenance. There also exists tools for automating the environment set-up process, such as Ansible or Vagrant, which create a virtual environment in which to do development and test activities, in such a way that environments that are created with the same script are as similar as possible (Clark et al., 2014). If the environments are standardized by one professional and then used and re-used then the implementation and maintenance of the environments gets easier and cheaper.

By far the biggest variable cost in test automation is the **creation of the test cases**. In manual testing the test case is created (designed) once, then repeated without much change. In automation the test case is designed and then implemented (the development part). The creation of the tests cases in automation is by far more laborious than with manual tests, and the

Factor	Definition	Author(s)
Test case implementation	Creating automated test cases can be compared to software development in means of effort, which is a lot more than in manual testing.	(Dustin et al., 2009; Hayduk, 2009; Hoffman, 1999; Ramler and Wolfmaier, 2006; Münch et al., 2012)
Test case maintenance	Automated test cases require more maintenance, since often when code is changed the tests need to be modified as well.	(Dustin et al., 2009; Hoffman, 1999; Ramler and Wolfmaier, 2006; Adams, 2002; Münch et al., 2012)
Test Execution	Test execution means executing the automated test scripts.	(Dustin et al., 2009; Hoffman, 1999; Adams, 2002; Münch et al., 2012)
Test results analysis	Automated tests can generate significant amounts of test logs, which explain the reasons the tests succeeded or failed.	(Dustin et al., 2009; Hoffman, 1999)
Personnel considerations	Using automation might require different skill sets, or a different amount of test engineers.	(Dustin et al., 2009; Adams, 2002)

Table 3.5: Variable costs of test automation.

investment in the automated tests is upfront. Additionally, the automated test cases are often either derived from manual test cases mid project, when it causes significant delays during the change process (Hoffman, 1999), or the test cases are first done once manually, then automated (Koirala and Sheikh, 2009). This means that the costs of creating manual tests are often included in automated tests, so they can be considered at least as expensive to create, in the best case scenario.

Related to test creation, the **maintenance** of the automated tests is usually a big expense. With manual tests, the tests are rarely changed, and changing them requires only minor redesign work. Automated tests are care-

fully scripted and if the implementation of the program changes, the tests need to be adjusted accordingly, unless made robust or to test high level components (i.e. unit tests automation vs. component level automation).

So far it seems that automated tests only bring more costs to the project. The real benefit, however, lies in the **test execution**. Executing automated tests is far quicker than executing manual tests. In reality, for example unit tests can be run automatically whenever there are changes to the code, which means running them requires no work from the developer or tester. However, many tests still need to be run manually, but the time it takes to execute the script is usually measurable in seconds. The benefit thus comes from being able to run tests more often, increasing confidence in the system.

Tests results **analysis** can take a long time in testing when done manually, i.e. comparing test output to predefined result criteria. Automated tests can also create a large amount of output, but it can also be used to parse and compile test results into a more readable form. Expected test results can be coded into the test script, compared against automatically and the output of the test can indicate, for example by color coding, the status of the test. In case of error, the resulting data can be compared to the desired output automatically, thus saving time of finding the desired data and output data, for example if an API test fails, showing the expected and resulted JSON string side by side for easy initial analysis of the problem.

Using test automation can also effect the **amount of test engineers** required. As stated in the fixed costs, training and implementation might require additional staff, but when the testing is well under way, tester needs might decrease since running the tests cases becomes faster, releasing testers to other activities, or to create more valuable or extensive tests.

Some factors that usually remain the same regardless of the testing method used are base test planning, base test design, defect reporting and reporting the results to management. These factors should be left out of the calculations when comparing manual and automated tests. (Hoffman, 1999)

3.3.2.2 Intangible Factors

Intangible factors, i.e. factors that cannot directly be measured in monetary (or other quantifiable unit) are difficult to incorporate into RoI calculations. Even so, the factors may contribute to a major part of the return of investment, even if it cannot be quantified (Hoffman, 1999) this brings an obvious

problem to benefit calculations, which will be tackled later in chapter 3.3.

There are a number of intangible benefits of software test automation in literature. First, automation allows the running of tests that would not be run manually (Fewster and Graham, 1999). Automation enables testing labour intensive, monotonous or boring tests that if ran manually over and over might decrease work motivation and cause decreases in work performance(Fewster and Graham, 1999). Also tests that require entering large amounts of data, or for example stress testing, that requires a large amount of repetitions is better done with automation (or might even be impossible to do feasibly using manual testing) (Rantanen, 2010).

Another benefit of test automation is that the tests will be run consistently. This means that they are run in the same way every time, reducing the error of human testers that might lead to incorrect results of the test (Fewster and Graham, 1999). The tests can also be run in the exact same way in different environments, allowing testing with other operating systems, databases etc (Fewster and Graham, 1999).

Test automation is considered to be software development in its own right, not just testing (Dustin et al., 2009). This means that it is more challenging and rewarding than manual testing (Hayduk, 2009; Fewster and Graham, 1999). Also, learning new technologies and techniques increases the professionalism of the (test) organization, further increasing motivation and enabling more advanced test case creation (Hoffman, 1999).

In terms of project schedule (which should be considered a tangible factor on its own), some factors have to be included here, due to their intangible nature. Firstly, if test automation is introduced mid-project the perceived effectiveness of the test organization decreases, since automation needs to be installed and implemented (Hoffman, 1999). While this delay in implementation is a tangible effect, the perceived decline in effectiveness can cause decrease in motivation (Fewster and Graham, 1999). Second, running automated tests can both increase confidence in the system, and thus allow a faster time-to-market, but also create a false sense of security (Fewster and Graham, 1999). The false sense of security comes from seeing all tests pass, but not considering that maybe the tests are inadequate, incomplete or not comprehensive enough (Fewster and Graham, 1999).

3.3.2.3 Simple Model For RoI

There already exists models of calculating RoI of software test automation in literature. The simplest way of calculating test automation RoI is to take the benefits of software test automation and compare them to the cost. Examples of costs are for example, but not limited to: hardware, software licenses, training and time to produce the test scripts (Kelly, 2004). The benefits are calculated from comparing the costs of automated testing and manual testing over a fixed period of time (Kelly, 2004).

Fewster and Graham (Fewster and Graham, 1999) present an example of calulating RoI for an automation project. Their example is based on the benefits gained from applying automation to testing:

$$\begin{aligned} RoI &= \frac{benefit - investment}{investment} \\ &= \frac{(C_m - C_a) * n_r - I_a}{I_a} \end{aligned}$$

where

Variable	Definition	Components
C_m	Cost of manual tests per iteration	Cost to design tests + cost to execute test cycle
C_a	Cost of automated tests per iteration	Cost to design tests + cost to execute tests
n_r	Amount of iterations (or releases)	The amount of iterations examined, can also be 1 for the total project.
I_a	Investment in est automation	Cost of testing tool + Cost to implement automated tests

This model, however, does not account for the maintaining of the tests that have already been implemented. Often, the tests need to be changed if the program under test changes significantly, i.e. the requirements change. Hoffman (Hoffman, 1999) presents a model for calculating RoI which accounts also for the maintenance tasks of the automated tests and also is more generally applicable over an arbitrary time frame:

$$RoI_a(\text{Over a time of t}) = \frac{\Delta\text{Benefits of Automation Over Manual}}{\Delta\text{Costs of Automation Over Manual}} = \frac{\Delta B_a}{\Delta C_a}$$

where

$$\begin{aligned}\Delta B_a(t) = &\Sigma\,(\text{improvement in fixed costs of automated testing})\,\textbf{*}\,(\textbf{t / Useful life}))\,+\\ &\Sigma\,(\text{variable costs of running manual tests }\textbf{a}\text{ times during time }\textbf{t})\,-\\ &\Sigma\,(\text{variable costs of running automated tests }\textbf{b}\text{ times during time }\textbf{t})\end{aligned}$$

$$\begin{aligned}\Delta C_a(t) = &\Sigma\,(\text{increased fixed costs of automation})\,\textbf{*}\,(\textbf{t / Useful life}))\,+\\ &\Sigma\,(\text{variable costs of creating automated tests})\,-\\ &\Sigma\,(\text{variable costs of creating manual tests})\,+\\ &\Sigma\,(\text{variable costs of maintaining automated tests during time }\textbf{t})\end{aligned}$$

This simple model, however has flaws that prevent it from being accurate enough. First, manual and automated testing cannot be compared as is, since they do not provide the same information about the system under test as automated tests are usually simpler they find different bugs (Kelly, 2004), mostly due to the different approach in the test usage (automated tests are usually built for regression testing, while manual tests are exploratory (Ramler and Wolfmaier, 2006)) and that automated tests cannot be used to execute all of the same tests as manual tests, ans vice versa (Kelly, 2004; Rantanen, 2010).

Another problem with simple RoI models is that often they do not take intangible factors into account (Rantanen, 2010), and they are in many cases impossible to take into account (Kelly, 2004). Intangible factors are hard to quantify, and thus often left out, especially in financial calculations where there usually is no room for a "human" component of cost of benefit. Nevertheless, many of the intangible benefits listed in the previous section should be taken into account in some way since they effect motivation, performance and other factors that indirectly contribute to the effectiveness of the whole organization.

3.3.2.4 A More Accurate Model For RoI

Since intangible factors can have a significant impact on the software project, they should be taken into account in some way. In his thesis, Rantanen (Rantanen, 2010) suggests a model by Kelly (2004) to calculating the intangible benefits of test automation. In the formula presented in the previous

section, the benefits are calculated as a sum of all quantifiable benefits. The addition to this calculation is to give all intangible benefits should be given a score between 1 and 5. Then the tangible benefits should be scaled with a number that makes the individual benefits be less or equal to 5. For example if benefits are 5000€ savings in test execution and 10000€ savings in tester salaries, then the scale would be 2000, making the amounts 2.5 and 5. After the scale has been determined, the costs should also be scaled with the same number to make them fit in the same range. (Rantanen, 2010)

The formula by Rantanen (Rantanen, 2010) (number 2000 used as an example for the scaling value):

$$RoI_a = \frac{\Sigma\,(\text{Value of intangible benefit}) + \Sigma\,(\text{Value of tangible benefit / 2000})}{\Sigma\,(\text{Cost / 2000})}$$

However, a clear problem with this model is, that it assumes that tangible and intangible factors are of the same scale, and that the tangible benefits and costs can be linearly scaled to match the intangible benefits. While the scaling of tangible factors allows for a quantifiable measure for RoI, accounting for all factors, the scaling procedure cannot be assumed to be accurate enough. Still, it can be used as an alternative measure with the formula from the previous section. For this reason, the intangible benefits should be used as more of a tie breaker, for example if RoI is estimated or calculated to be close to 1, the intangible benefits should be considered to turn the RoI into a positive number.

3.3.3 Net Present Value

Although many software projects don't last longer than a year or two, the issue of net present value (NPV) should be raised when assessing the business impacts of test automation. NPV is a way of calculating discounted cash flows, which means that due to inflation, a dollar received 1 year from now is worth less than 1 dollar today (Horngren et al., 2012). In short projects this is irrelevant, but in larger projects, or when estimating automation gains in the future, or calculating them from the past, the amounts should be discounted into present monetary values to avoid miscalculation and bad decisions based on the assumption that money maintains its value through time. For example, with an inflation rate of 3%, a 10000€ cash flow 5 years from now would mean:

$$\frac{10000€}{1.03^5} = 8600€$$

NPV can also be used to calculate the value of the investment compared to other means than inflation: opportunity cost (Horngren et al., 2012). If the money, instead of being invested in test automation, can be invested in something else, for let's say, at an expected 10% interest, then the same formula can be used to calculate the net present value of the same 10000€ return in 5 years by substituting the inflation percentage with the obtainable interest from another investment (Horngren et al., 2012). The opportunity cost here is *"not having the money today to invest in something else"*. The previous example with the 10% interest rate is 10000€ in 5 years is 6200€.

3.4 Chapter Summary

In this chapter the basic principles of testing, test automation and the benefits and costs of test automation have been discussed based on a literature review. Based on literature, test automation is an alternative to manual testing, either to replace it or to be used in conjunction. While test automation promises many benefits, it can still be a risky investment, and its potential benefits should be assessed carefully. This assessment should include a cost-benefit analysis of test automation.

The most common way of assessing test automation effectiveness is to calculate its return on investment. These calculations, however, rarely take into account the intangible benefits and costs of test automation, which are hard to quantify. The qualitative factors should then be either incorporated to the calclations with a proper scaling system, or then be used as a qualitative *"tie-breaker"* in cases when the benefits and costs of automation are close to being equal.

Chapter 4

Results

This section presents the results of the case study. The results analysis consists of analysing the RoI factors of and the interview results of the case. The RoI factors analysis contains calculations made on its efficiency based on figures received from the customer and the company implementing the automation project.

4.1 RoI Calculations

As discussed in chapter 3, return on investment consists of analysing the costs and benefits of the investment. The case study project presented in this research has been under way for 17 months (at the time of writing this thesis), which is 2 full iterations and one partial iteration. Each iteration lasts for 6 months, containing a 9 week testing period. Since the third iteration was almost complete (one out of six months left) at the time of writing this thesis, the estimated numbers for third iteration testing period are used, which at this point are fairly accurate since the iteration is nearly complete.

In the case company, some of the metrics described in tables 3.4 on page 38 and 3.5 on page 40 were not applicable. For example, manual test case design and implementation were considered to be a single activity, and thus could not be clearly separated. Nevertheless, the costs were analysed as a single cost factor. For this reason, the simpler formula for RoI by Kelly (2004) was used for the calculations instead of the more specific ones by Fewster and Graham (1999) or Hoffman (1999). The main distinction is, that the two latter formulae separate the costs and benefits of automation and manual testing in the formula, while the formula by Kelly only lays out the basics, allowing the costs and benefits to be calculated as best fits the

case. The factors listed in tables 3.4 and 3.5 are used as a baseline for the analysis.

4.1.1 Automation costs

Starting with the costs of the automation project, the **total costs** are presented in the same form as in tables 3.4 and 3.5. All the cost factors are explained with detailed calculations below, and then multiplied by the amount of iterations (3) to be presented in table 4.1. For the purpose of this case, test implementation and design and execution and analysis were combined to one cost factor. The combinations were made since for manual testing, the total cost of design and implementation and of execution and analysis were known, but not separately. Thus, they were combined in both manual and automated costs to give a better comparison of the two.

Factor	**Amount**	**% Of Total**
Cost of hardware	8000€	3.2
Licences for tools and testware	Negligible	-
Tool training	Included in other activities	-
Test automation environment set-up and maintenance	3000	1.2
Test case implementation & design	160380€	66
Test case maintenance	18000€	7.4
Test Execution & analysis	53820€	22
Personnel considerations	0€	0
Total	**243200€**	

Table 4.1: Total costs of automation case.

Hardware costs consisted of a single test server. The server specifications were kept confidential, but the cost of the hardware was estimated at 8000€ by the IT specialist of the company. The cost was an estimate due to modifications made during the project.

Licences for tools and testware were considered negligible, since they only consist of a single Windows licence, which was a minor cost in the scope

of the project. All other tools used were open source frameworks with no licence or other purchase fees.

Tool training was not considered a separate cost item since most automation developers were already familiar with the technologies, and those who were not were *educated on the job*" without any specific, separately organized training.

Test environment set-up was calculated to be a two-day exercise, costing 1200€. This consisted of installing all required software and tools and configuring the system according to the needs of the automation project. **Test environment maintenance** was a 1-2 day job in each iteration, so far costing approximately 1800€ (this number is explained with allocation calculations below). Maintenance consisted of reconfiguring the environment when tests are modified and fixing errors in any configuration.

Test case implementation The calculation for test case implementation had to be separated into two areas: during test cycle, and outside test cycle since the allocations of work for each activity were not the same in both cycles. The costs for each **iteration** test case implementation were calculated as follows:

$$
\begin{aligned}
A_{it} &= \text{Implementation during test cycle * test cycle weeks * weekly cost of automation consultants} \\
&= 0.5 * 9 * (5\text{d} * 7.5\text{h/d} * 80€/\text{h}) \\
&= \mathbf{13500€} \\
A_{id} &= \text{Implementation during development cycle * test cycle weeks * weekly cost of automation consultants} \\
&= 0.9 * 14.8 * (5\text{d} * 7.5\text{h/d} * 80€/\text{h}) \\
&= \mathbf{39960€} \\
A_i &= \text{Automation implementation total per iteration} \\
&= A_{it} + A_{id} \\
&= \mathbf{53460€}
\end{aligned}
$$

With similar logic as in the automation implementation formula $(\mathbf{A_t})$, the costs for automated tests **execution**, **maintenance** and **environment maintenance** were calculated for **each iteration**. Execution of tests are is also split into test and development cycles, while test and environment maintenance are a *per-iteration* task.

A_{et} = Execution during test cycle * test cycle weeks * weekly cost of automation consultants
= 0.5 * 9 * (5d * 7.5h/d * 80€/h)
= **13500€**

A_{ed} = Execution during development cycle * test cycle weeks * weekly cost of automation consultants
= 0.1 * 14.8 * (5d * 7.5h/d * 80€/h)
= **4440€**

A_e = Automation execution total per iteration
= A_{et}+A_{ed}
= **17940€**

A_{mt} = Test maintenance/iteration * cost of automation consultants
= 2w * (5d * 7.5h/d * 80€/h)
= **6000€**

A_{me} = Env maintenance/iteration * weekly cost of automation consultants
= 1d * (7.5h/d * 80€/h)
= **600€**

Although the iterations may vary in work loads, they were so long that using the same values of work allocation, as reported by the consultants who implemented the automation project, were sufficiently accurate.

Finally, **personnel** considerations. While approximately 10% of test cases in the case company were being run automatically, the number of testers in the organization has not diminished, but their time was freed to other activities and projects. This freed time can be regarded as *saved money* since no extra personnel need to be hired for testing activities.

4.1.2 Manual Testing Costs

To make an even comparison with automated test costs, the same variables for manual testing will be calculated and explained. Starting with a similar table as table 4.1, the costs of manual testing during the automation project are separated in table 4.2. The cost of **hardware**, **licences**, **training**, **and environments** in the manual testing project are all either zero or negligible compared to the size of the testing process. The numbers are negligible since

most of the hardware, for example, was already there for the project.

Factor	**Amount**	**% Of Total**
Cost of hardware	0€	0
Licences for tools and testware	0€	0
Tool training	0€	0
Test automation environment implementation and maintenance	0€	0
Test case implementation & design	191450€	22
Test case maintenance	Included in implementation and execution	-
Test Execution & analysis	681140€	78
Personnel considerations	0€	0
Total	**872590€**	-

Table 4.2: Total costs of manual testing during automation case.

Test case implementation and design was conducted by 12 designers, who worked on creating tests for this project on a 20% work allocation. The salaries of test designers were not revealed, but an industry standard of 4500€ was used for the calculations:

$$\begin{aligned} M_d = &\text{ Amount of designers * 26 weeks/iteration * allocation * Salary} \\ &= 12 * 26\text{w} * 0.2 * (5 \ / \ 22 * 4500€) \\ &= 63800€ \end{aligned}$$

Test case maintenance was included in the execution and design tasks since the tests are quite robust and flexible, and hardly need ever modifications. Still, the cost was more than zero, and easily included in the design and execution tasks, since they are often fixed on the fly.

Test execution costs were calculated with a similar calculation as with the automated tests. The testing organization has 20 manual testers and 20 integration testers (who also test manually). During the 9 week test cycle

the manual testers work on testing with full allocation, while the integration testers work with 50% allocation. Outside the test cycle the testers do not work on the project at all. Manual testing cost **per iteration** is:

$$
\begin{aligned}
M_t = &\text{Amount of manual testers * Amount of integration testers *} \\
&\text{allocation * 9 weeks * salary} \\
= &20 * (20 * 0.5) * 9 * (5 / 22 * 3700€ \\
= &\mathbf{227000}
\end{aligned}
$$

4.1.3 Scaling effort and project Roi

Having calculated the costs of both manual and automated testing, there was one more consideration to make before calculating the RoI of the project: during the project the same amount of automated test cases were not executed and created as manual tests. Different amounts of manual tests were executed in the iterations, and both manual and automated tests were created during iterations. The summary of the difference (per iteration) is presented in table 4.3:

	Manual	**Auto**	**Ratio**
Tests executed in iteration 1	255	26	0.102
Tests executed in iteration 2	385	43	0.112
Tests executed in iteration 3	392	97	0.247
Tests implemented in iteration 1	33	26	0.788
Tests implemented in iteration 2	33	17	0.515
Tests implemented in iteration 3	33	54	1.636

Table 4.3: Automation and manual tests execution and implementation differences.

The reason for listing these values and ratios is to scale the effort of automation creation and execution to the same scale as with manual testing. For example, if 10000€ was used to create 10 manual tests, and 5 automated

tests were created, the cost of manual testing for the same amount of tests would be only 5000€, since only half the amount of tests were automated. Obviously all tests cannot be compared interchangeably (e.g. long and short tests), but differences in these will cancel each other out in average in the long run. Additionally, the tests implemented and executed were fairly similar according to the testers. The scaling of costs per iteration and totals are presented in table 4.4. The column **manual scaled** is calculated by taking the **ratio** column from table 4.3 and multiplying the corresponding manual cost by it. For example, in iteration 1 manual testing costed 227000€, but only 10.1% were automated, so the scaled manual cost is 227000 * 0.101 = 23100€.

	Manual	**Auto**	**Manual scaled**	**Automation benefit**
Execution cost in iteration 1	227000	17940	23100	5200
Execution cost in iteration 2	227000	17940	25300	7400
Execution cost in iteration 3	227000	17940	56100	38200
Total	**681100**	**53820**	**104600**	**50870**
Implementation cost in iteration 1	63800	53460	50280	−3180
Implementation cost in iteration 2	63800	53460	32800	−20580
Implementation cost in iteration 3	63800	53460	104400	50979
Total	**191450**	**160380**	**187580**	**27200**

Table 4.4: Automation and manual tests costs and scaling.

Having calculated and scaled all the costs of both manual and automated testing, we can apply the formula for RoI by Kelly (2004):

$$RoI = \frac{Benefit}{Investment} = \frac{\text{Cost of manual testing} - \text{Cost of automated testing}}{Investment}$$

Table 4.5 presents the total costs for each iteration, and the project total cost, until iteration 3. As the costs for test and environment maintenance for

manual testing are considered to be 0, the corresponding values for automation are added of the implementation and execution costs in the calculations without any further examination. Note, that the setup costs of the automation project (hardware, 8000€ and environment setup, 1200€) are included in the costs of the first iteration. This makes the first iteration's automation costs 78000 + 9200 = 87200€.

	Manual	**Auto**
Total cost for iteration 1	73430	87200
Total cost for iteration 2	58230	78000
Total cost for iteration 3	160600	78000
Total cost for project	**292270**	**243200**

Table 4.5: Automation and manual tests costs and scaling.

Table 4.6 summarizes the RoI for all iterations and the project until iteration 3. In this case, the *investment* used in the formula is the cost of automation, since the customer company invested in a full service automation project from an external provider.

Manual	**Auto**
RoI for iteration 1	−0.158
Roi for iteration 2	−0.253
Roi for iteration 3	1.059
Total Roi for project	**0.201**

Table 4.6: Automation and manual tests costs and scaling.

4.2 Interview Analysis

This section covers the results of the interviews. The section is divided into subsections for benefits, pitfalls and other considerations of test automation. The analysis is carried out by listing benefits and pitfalls of automated testing and how they were perceived by the interviewees.

4.2.1 Test Automation Benefits

Table 4.7 summarizes the main benefits listed in the interview results and difference between the benefits listed in 3.2 in the literature review section.

In the table **X** denotes that the benefit is often mentioned in literature or the interview results and **x** that is was mentioned less frequently.

Benefit	Literature	Interviews
Faster	**X**	**X**
Attention	**X**	**x**
Consistency	**x**	**x**
Performance	**X**	
Regression tests	**x**	**x**
Opens communication		**X**
Frees resources	**x**	**X**

Table 4.7: Benefits of test automation.

The notion that automated testing is **faster** than manual testing was brought up by all interviewees. The part of the process where the increase in speed occurred varied. The speed increases were seen in the detection and reporting of tests, in the speed of execution or even an in the overall deployment cycle. Two of the interviewees stated that when the tests that have been automated can be instantly run, and the errors which were mainly generated by changes in the software are discovered more quickly they can also be fixed more quickly. Another two simply said that the execution of automated is plainly faster than manual tests, which results in quicker regression testing.

Attention span of the testers was brought up by two of the interviewees. The manual testing process is very repeatable work, and in time the tester will get bored or numb of doing the same thing over and over again:

> "Automated testing is clearly more reliable since the tests are always executed in the same way. People often make human errors and get tired of repeating the same routines over and over again."

In the discussion about attention, **consistency** was brought up by one of the interviewees: the tests are only consistent (i.e. they can be relied upon to execute the same thing every time) if the system can be relied upon and is stable. The same point was brought up by others, one of whom made a distinction between manual and automated testing considering consistency:

> "Manual testing lets the tester be creative, which can be both good and bad: good in the sense that the testing can be different

> (expllorative) and bad since the actual thing to be tested might not get tested whereas automated tests execute exactly what they are told to."

The **regression** benefit of test automation was brought up both in the perspective of easier integration testing as well as developing new features. Integration testing can be difficult when testing interfaces between systems. If something fails in the other end of the test, it can be difficult to access manually, but easier with automation. When developing new features, running automated integration tests on new changes immediately after they are made lowers the workload of testing and rework in the case of extensive testing periods instead of automation.

Two most interesting benefits in the interviews, hardly mentioned in literature, were **communication** and freeing up resources. One of the interviewees pointed out that automation simplifies the discussion between business and development activities. He continued about communication:

> "The biggest benefit of automation is that things that used to rely on person-to-person communication, for example: are we satisfied with our testing?, become relatively yes-or-no answers... At the customer organization there has long been a problem of knowing that testing is not in a satisfactory level, but the means of discussing the issues are not there, since he supplier can easily explain the problems in the context. Automation takes the discussion and the unfortunate tendency of blurring facts out of the equation, and the question becomes a yes-no-answer."

Three of the respondents brought up the communication with the provider of the software. First, when the supplier tells that something in the software has been fixed, the tests can be instantly run if this truly is so. Continuing from the first point is that when the feedback is quicker to the supplier, the bugs can be fixed quicker, and then retested after the fixes are made. The feedback loop is quickened by being able to immediately send the bug reports (or during the first day, as compared to at least three days in the case company).

Finally, **freeing up resources** was considered a significant benefit of test automation:

> "The people allocated to do manual testing have primary work tasks, which would consume all their work time if possible, which

> means that the resources are wasted. They (the case company) don't have any dedicated manual testers, so the ones testing manually have to disregard their actual work responsibilities when allocating them to do manual testing."

Another interviewee had an intriguing point that if there is a bug in the system, no one had to wait for the *"now is has been fixed"* -signal, but the tests can be run automatically and the person can focus on something else. Then the test will in time either pass or the message "it has been fixed now" will arrive and the tester only needs to check the test process if it passes or not, having to do no idle waiting meanwhile and being able to do more productive work while waiting.

4.2.2 Test Automation Pitfalls

Table 4.8 summarizes the main pitfalls listed in the interview results and difference between the benefits listed in 3.3 in the literature review section. In the table **X** denotes that the benefit is often mentioned in literature or the interview results and **x** that is was mentioned less frequently.

Pitfall	Literature	Interviews
Initial effort	**X**	**X**
More maintenance	**X**	**X**
Volatile environments		**X**
Finding new bugs	**X**	**x**
Cannot replace humans	**X**	**x**

Table 4.8: Pitfalls of automated testing.

The **initial effort** of automation was a concern almost every interviewee brought up. The answers were divided amongst the design and implementation of the test cases and the design and implementation of the automation system:

> "Manual testing is, of course, faster to ramp up, but it isn't sustainable in the same way, since the same resource has to be spent every time if the same test has to be run again."

> "The initial effort (of automated testing) can be difficult to justify because implementing automaton takes more time than one manual testing period. However, if there are many testing periods it will pay itself back."

The respondent who had been implementing automation the longest in the case company said that on average they can create one automated test per day, on a good day two. The case company had calculated that on average a tester can run 7 manual tests per day. This suggests that automation would pay itself back in approximately 7 iterations of tests.

The design and implementation of the test cases in the case company was especially difficult, since the organization was mostly in silos of people and job positions making the finding of the correct person to talk to about a specific test was difficult. Often the design would go on iteratively, using the persons available at the time, then moving on and coming back when they are available again. Last interesting point about the implementation of the test cases was that the timing of implementation, i.e. when to create the test case, is a crucial factor in test design:

> "The biggest drawback in test automation in my opinion is that it is very hard to say when to implement the test cases. That is, when is the system stable enough that we can be sure there will be no more, or just little changes to the system. If the parts of the system change often, there will be a lot of overhead with automation, since if has to be adjusted every time."

This quote also raises the problem of **maintenance**. All respondents stated that automation takes more maintenance than manual testing. Mostly the problem was seen to be in the timing of the automation implementation and that the frequent changes in the software cause significant overhead in automated test maintenance:

> "If the environment changes in a rapid pace, automation maintenance can quickly become expensive. Automation should be designed so that if the user interface is going to change in the near future, automation might be useless to implement."

One point raised up was that most customers are often surprised that automation needs maintenance. In many cases, they expect that if the tests have been implemented once, it is enough, and no further work is required.

Closely related to maintenance, **volatile environments** were also a point raised by almost all respondents. While it may seem that they are the same thing, the need for maintenance is inherent to automation when changes to the system are made, but volatile environments can cause headaches in the execution phase and result analysis.

> "Another significant challenge is that when environments and software are unstable, the automated test will run the same operation twice exactly the same, but the result may still differ. This raises to question whether the test is broken or the application?"

In the answers it was often pointed out that the test environment for automation should be separated from other testing. This is due to the sensitive nature of automated tests, and that changes to the test environment and the system in the environment will easily cause breakdown of the tests. Frequent updates to the system under test might also cause problems in the implementation of the test cases.

Two of the respondents noted that test automation is not as good in **finding new bugs** as manual testing. One of them noted that in a best case scenario, automation can free up manual testers to do explorative testing instead of regression testing.

> "Most bugs are found with explorative testing, not regression testing. This means, that if a large automation project is carried out, what the result will be is a set of automated regression tests."

The drawbacks of more maintenance, volatile environments and finding new bugs are all connected to the fact that automation **cannot replace human testers**.

> "And experienced manual tester can mitigate changes in the system. For example if the test is to create a new user, the tester will know how to do it, even if the layout changes. For automation, it is always a field, a drowpdown menu or a button. If these change, the automated test will fail, while the manual tester can continue regardless of the change."

> "For example, if a field has changing IDs or they are missing completely (WEB site testing), it can be very difficult to automate... If an ID changes, a manual tester won't care. If the font changes, the manual tester can mitigate the change."

Mainly these quotes tell us that manual testing is less strict automated testing. The script that was automated relies on the *exact state* of elements and functionality, while the manual tester only needs to care about the *process* they need to test.

4.2.3 Other considerations

In the interviews one more important factor rose among pitfalls and benefits: automation strategy; in other words, *when to automate*. When to automate is a question that should be asked as soon as possile in the automation project. This point is relevant to the volatile environments problem mentioned before. It means, that choosing to automate tests in mature parts of the software can drastically reduce the amount needed for maintenance:

> "Maintenance of automated tests can be reduced if the time when automation is implemented is chosen carefully. If not, a lot of extra work may be required."

From the interviews it was clear that the relatively low costs of maintenance in the project resulted from a good strategy in choosing the point in time in which to automate.

Chapter 5

Discussion

5.1 RQ1: How can test automation effectiveness be measured?

The literature research conducted in this thesis suggests that the most common way of measuring test automation effectiveness is to measure its return on investment (RoI). Measuring RoI is not always simple and it contains various elements that can be counted as benefits or costs for test automation. A few sources also suggest tracking metrics, such as test coverage, but in truth test coverage does not tell much about the effectiveness of testing since it only explains how many lines of source code have been run by testing. Coverage analysis is usually limited to line coverage without regard to logic coverage. Additionally, in business the final determining factor of effectiveness is money, which also prompts for using RoI metrics in measuring effectiveness. Furthermore, most other metrics can be transposed to monetary benefit, by for example work time allocation and work hour cost.

Usually RoI is measured by comparing the cost of doing the same amount of testing or the same testing activities in manual testing against doing it automatically. In the case that no manual testing has already been done, the comparison would be done by counting the costs of automated testing and comparing them to the risk-cost of not testing at all. In other words, the benefit is calculated from *"How much money do we use to do this manually"* minus *"How much money do we use to do this automatically"*.

Some of the costs of automation include hardware costs, licence costs, tool training, test case implementation and maintenance and test results analysis. These costs are only a summary of all possible costs that can be

used, but they present the majority of the costs that usually occur and that occurred in the case study company. In addition to these, any cost that can be attributed to manual and automated testing, but that is not a cost that will occur no matter the testing method (for example test design, which is usually neglected since it is similar to both manual and automated testing), can be used in the calculation.

The benefits of automation are often represented as either the money saved compared to using manual testing, or the tradeoff of not testing at all and facing the risks. The comparison of manual and automated testing is done by calculating the same costs for manual testing as for automated testing, and finding the difference. After the costs of manual and automated testing are calculated, the formula for RoI can be applied. Extended from the basic formula of RoI, a version of the common RoI formula was used in this thesis, which explains the logic of finding the benefit and cost of automation:

$$RoI = \frac{Benefit}{Investment} = \frac{\Sigma(\text{Cost of manual testing} - \text{Cost of automated testing})}{Investment}$$

The sum in the nominator means calculating the difference between manual and automated testing for all cost factors. For example (cost of manual implementation - cost of automated implementation). The cost of manual testing consists of manual test design, implementation, and test execution and reporting. The cost of automated testing consists of the design, implementation, execution, reporting, hardware costs, software licences and training. In this case, the cost of the investment is the cost of automation.

The question of automation effectiveness measurement is answered in this thesis based on a literature review of multiple sources. The amount of sources and the consistency of the methods bring sufficient grounds for the results, but the area of non-business metrics (such as test coverage, defect rates etc.) is left out of scope, which leaves room for further research. Compared to previous research, this thesis more thoroughly collects different cost factors of testing and different methods of calculation the RoI of automation, which allows for more synthesized view of financial effectiveness assessments.

5.2 RQ2: What is the impact of test automation to software project profitability?

In this thesis, the impact of test automation to software business was investigated by conducting a case study on the business information retrieved from the case company and by conducting interviews on the test automation consultants implementing the automation project. The first part of the case study was to find the impact the automation project had on the RoI of testing. In the research, the costs of both automation and manual testing were calculated from actual sources inside the case company (customer), and from sources in the implementing consultant company. The first source consisted of calculations made by the customer, and the latter by interviewing (not connected to the interview study) the consultants implementing the automation project and from the project data archive. Since automation is still only a fraction of total testing, the costs of manual testing were scaled down to match the same amount of coverage as the automated tests. In other words, 10% of the tests were automated, so the costs of manual testing were scaled down to the point where only 10% of testing was manual.

The results from the RoI calculations in the results chapter show that automation gave a benefit in the testing effort in the case project, and by the time of this research (after 18 months of automation) was approximately 0.20. This means a 20% profit after investment has been covered by earnings. In the data presented by the customer, it was evident that the effectiveness of automation was increasing, and the RoI would most likely increase during the next one or two iterations.

This thesis only focused on one large case, and thus the accuracy of the result is not very high. Nevertheless, the project has been running for 18 months, with dozens of testers involved, which increases the scale of the research. If the same calculations were to be made on several smaller projects, the results would be more accurate in means of variance, but the scale of the operations are different, and thus hardly comparable.

The data used in the calculations was collected from credible sources in both the customer company and the implementing company. However, most of the data was not documented in a specific way, and many of the costs of automated and manual tests were professional estimates. Still, the total costs of both projects were known fairly accurately, so only the division of costs was estimated, not the total costs itself.

5.3 RQ3: What are the benefits and pitfalls of automated testing?

Finally, this research attempted synthesize the benefits and pitfalls of test automation. Both the literature review and the second part of the case study, the interviews, found several benefits and pitfalls.

The interviews had two main goals: to find benefits and pitfalls of test automation and to discuss the relevant financial implications of test automation. As none of the interviewees had any business background, most of them had no say on the effectiveness of test automation in a strictly business sense. For this reason the first goal was satisfied better than the latter one and so the pitfalls and benefits of test automation were the main topic in all interviews.

The results of the interviews were very similar to the results from the literature review, and most of the points found in the interviews were also present in literature. However, all points in literature were not brought up in the interviews. Still, the interviews presented a strong empirical evidence to the benefits and pitfalls of automated testing found in literature, and that even though some of the sources were quite old, the same fundamental ideas still applied.

The main benefits of test automation found in the interviews were: speed, meaning the ability to run tests much faster; attention span of testers, meaning the less monotonous work of developing automated test cases over executing the same manual tests over and over again; consistency, meaning the similar execution of each test case on every run; regression testing, meaning the re-running of old tests every time changes are made to ensure that new functionality or fixes do not break the old implementation; communication, meaning faster interaction between supplier and customer in case of errors in the software; and better use of resources, meaning developers being able to focus on more important tasks than just executing manual tests.

Speed, attention span of testers, consistency, regression tests and better use of resources are benefits verified by one or more literature resources, and thus seem to be valid benefits of test automation. Communication was not present in the literature review, which brings to question either the methods used in the case company, or even change in the testing practice, which has not yet made its way as part of the existing pool of knowledge.

The main pitfalls of automation in the interviews were: initial effort, meaning the comparatively large effort of creating tests compared to manual testing; maintenance, meaning the increased need of maintaining tests when using automation compared to manual testing; volatile environments, meaning the need of constant changes in automated tests if the environments or the system under test are not stable; less new bugs found, meaning that manual tests are more effective in discovering new bugs; and inability to replace humans, meaning that automation cannot mitigate situations that humans can when testing fails but should pass and vice versa.

Initial effort, maintenance needs, finding of new bugs and inability to replace humans were all often quoted in the literature review, and thus seem to be valid as automation pitfalls. The volatile environment situation was not present in literature, which in this case is most likely rooted in the organization of this project, where the customer, automation implementation and software implementation are all different entities.

The results of this thesis mainly complement existing literature on test automation benefits and pitfalls, but also both brings out new benefits and pitfalls. A literature review by Rafi et al. (2012) contains most of the benefits and pitfalls found in this research. Still, the question of *when to automate* and the benefit of improved communication are hardly present in current literature. In the interviews conducted in this research, these two factors were a topic with most of the interviewees, and thus seem to also be valid benefits of test automation, but require further research to be added into the current body of knowledge in test automation.

Judging from the results of the interviews and the literature review, the benefits and pitfalls of test automation can be seen as a valid starting point for analysis if an automation effort is to be undertaken. Many benefits and pitfalls were left out of the results either because they were not a significant monetary investment or drawback, or because they were rarely in the interviews or literature.

Chapter 6

Conclusions

The research problem of this thesis was: how does the implementation of test automation affect software projects? Judging from the results of the case study, the impact of test automation seems to have a positive impact on business by providing savings against manual testing. The results seem to indicate that the benefits of automation are realized after a longer period of time. Sometimes the benefits will not be realized even in the same project that automation is first applied, but rather in the long run.

In addition to monetary benefit, the study reveals intangible benefits that include running tests that could not be run manually, or that would require a tremendous effort, consistency of test runs and tester job satisfaction through challenge and reward. These (and other intangible) benefits and pitfalls cannot be quantified as such, and should be taken into account when assessing the profitability of automation. For instance, if RoI of the project is 0, or slightly negative, but the testers are very satisfied in their work, or if the tests require entering the same values ten thousand times (virtually impossible to do manually every day), then the project could still be seen as having a profitable outcome (i.e. RoI >0).

This thesis only covered a single case company. While the case was a large and a long project, the data gained from it is still only from the sources of the customer company and the implementing company. This leaves room for more thorough case research, to include more companies and projects, that might even be of differing size. Additionally, the benefits and pitfalls of automation could be studied further by a wider literature review and a more diverse set of interview respondents (i.e. different job descriptions etc.). Next, the possibility of analyzing effects on business could be studied from another point of view than just business (RoI), such as reviewing metrics,

code coverage or amount of new bugs during development in a longer time span. Finally, the intangible factors mentioned before present an area of study on their own, regarding effectiveness and how they could be transposed into a quantifiable metric that is easier to measure in terms of effectiveness.

Bibliography

Ieee standard glossary of software engineering terminology. *IEEE Std 610.12-1990*, pages 1–84, Dec 1990. doi: 10.1109/IEEESTD.1990.101064.

E. Adams. The business argument for investing in test automation. 2002. `http://www.ibm.com/developerworks/rational/library/content/RationalEdge/dec02/TestAutomation_TheRationalEdge_Dec2002.pdf`. Accessed 12.10.2015.

J. Bach. Test automation snake oil. In *Proceedings of the 14th International Conference and Exposition on Testing Computer Software (TCS'99)*, 1999.

J. Bach. Agile test automation. 2003. `http://satisfice.com/articles/agileauto-paper.pdf`. Accessed 22.9.2015.

P. Bourque and R. E. Fairley. *Guide to the Software Engineering Body of Knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press, 2014.

D. Clark, A. Culich, B. Hamlin, and R. Lovett. Bce: Berkeley's common scientific compute environment for research and education. In *Proceedings of the 13th Python in Science Conference (SciPy 2014)*, pages 5–13, 2014.

P. Corbetta. *Social Research: Theory, Methods and Techniques*. SAGE Publications, Ltd, 0 edition, 2003. ISBN 9781849209922.

E. Dustin, T. Garrett, and B. Gauf. *Implementing automated software testing: How to save time and lower costs while raising quality*. Pearson Education, 2009. ISBN 978-0-321-58051-1.

P. Eriksson and A. Kovalainen. SAGE Publications Ltd, 2008. ISBN 9781412903172.

M. Fewster and D. Graham. *Software test automation: effective use of test execution tools*. ACM Press/Addison-Wesley Publishing Co., 1999. ISBN 0-210-33140-3.

D. Graham and M. Fewster. *Experiences of test automation: case studies of software test automation*. Addison-Wesley Professional, 2012. ISBN 978-0-321-75406-6.

B. Hayduk. Maximizing roi and avoiding the pitfalls of test automation. 2009. `http://uktmf.com/sites/default/files/AidusMcVeighAvoidingPitfalls.pdf`. Accessed 25.9.2015.

D. Hoffman. Cost benefits analysis of test automation. *Software Testing, Analysis, and Review (STAR) Conference.*, 1999.

C. T. Horngren, S. M. Datar, and M. V. Rajan. *Cost Accounting: A Managerial Emphasis*. Pearson Education Limited, 2012. ISBN 978-0-273-75387-2.

J. Itkonen, M. V. Mäntylä, and C. Lassenius. Defect detection efficiency: Test case based vs. exploratory testing. In *Proceedings of the First International Symposium on Empirical Software Engineering and Measurement, ESEM 2007*, pages 61–70. IEEE, 2007.

K. Karhu, T. Repo, O. Taipale, and K. Smolander. Empirical observations on software testing automation. In *Proceedings of the Second International Conference on Software Testing Verification and Validation (ICST 2009)*, pages 201–209. IEEE, 2009.

M. Kelly. The roi of test automation. In *Proceedings of the International Conference on Software Testing Analysis and Review (STAREast)*, 2004.

S. Koirala and S. Sheikh. *Software testing*. Jones & Bartlett Learning, 2009. ISBN 978-0-7637-8297-9.

P. Laukkanen. Data-driven and keyword-driven test automation frameworks. Master's thesis, Helsinki University of Technology, 2006.

Y. K. Malaiya. Automatic test software. *Wiley Encyclopedia of Electrical and Electronics Engineering*, 2011.

S. Münch, P. Brandstetter, K. Clevermann, O. Kieckhoefel, and E. Reiner Schäfer. The return on investment (roi) of test automation. *Pharmeutical Engineering*, 32(4):22–32, 2012.

G. J. Myers, C. Sandler, and T. Badgett. *The Art of Software Testing, 3rd Edition*. John wiley & sons, 2011. ISBN 978-1-118-03196-4.

D. M. Rafi, K. R. K. Moses, K. Petersen, and M. V. Mäntylä. Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. In *Proceedings of the 7th International Workshop on Automation of Software Test*, pages 36–42. IEEE Press, 2012.

R. Ramler and K. Wolfmaier. Economic perspectives in test automation: balancing automated and manual testing with opportunity cost. In *Proceedings of the 2006 international workshop on Automation of software test*, pages 85–91. ACM, 2006.

K. Rantanen. Benefits of automated system-level regression testing: A case study. Master's thesis, Aalto University School of Technology, 2010.

J. Ruusuvuori and L. Tiittula. *Haastattelu: tutkimus, tilanteet ja vuorovaikutus*. Osuuskunta Vastapaino, Tampere, 2005. ISBN 951-786-169-0. Translation: own.

M. NK. Saunders, M. Saunders, P. Lewis, and A. Thornhill. *Research methods for business students, 5th Edition*. Pearson Education India, 2011. ISBN 978-0-273-71686-0.

D. W. Turner III. Qualitative interview design: A practical guide for novice investigators. *The qualitative report*, 15(3):754–760, 2010.

J. A. Whittaker. *Exploratory Software Testing*. Addison-Wesley Professional, 2009. ISBN 978-0-321-63641-6.

K. Zallar. Practical experience in automated testing. *Methods and Tools*, 8 (1):2–9, 2000.

Chapter 7

Appendices

7.1 First Appendix: Interview Questions

This appendix contains the questions designed for the interviews. Boldfaced items were the main questions, asked from all participants, other were additional questions asked for elicitation or if there was sufficient time.

- **What was the motivation for the project?**
- **What was your part in the project?**
- **What was done in the project?**
- How was testing originally implemented in the project?
- **What was hoped to achieve by using test automation?**
- **How does test automation differ from manual testing?**
- What does implementing test automation require when migrating from manual testing?
- What does implementing test automation require when starting from the beginning of the project?
- **What benefits does automated testing have over manual testing?**
- **What drawbacks does manual testing have over automated testing?**
- What was the outcome of the project?

- How was the desired outcome achieved?
- How would you improve test automation practices in the project?