CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Automated Performance Regression Analysis: An Industrial Case Study

Master of Science Thesis in Software Engineering

Mikael Fagerström
Emre Emir İsmail

Department of Computer Science & Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2016

Supervisors: Dr. Eric Knauss, Dr. Patrizio Pelliccione
Examiner: Dr. Miroslaw Staron

Chalmers University of Technology
University of Gothenburg
Department of Computer Science & Engineering
SE-412 96 Gothenburg
Sweden
Telephone + 46 (0)31-772 1000

Acknowledgements

I would like to thank my thesis partner Mikael, our academic and industrial supervisors and our department manager.

I would like to dedicate all the work I put in this thesis work to my father who recently passed away.

Emre Emir İsmail, Gothenburg 20/11/2015

I would like to thank my thesis partner Emre, our academic and industrial supervisors as well as our manager.

I would also like to thank my family for being a source of strength and support.

Mikael Fagerström, Gothenburg 20/11/2015

Abstract

Along with the technological developments and increasing competition there is a major incentive for the companies to produce and market high quality products before their competitors. In order to conquer a bigger portion of the market share companies have to ensure the quality of the product in a shorter time frame. To accomplish this task companies try to automate their test processes as much as possible. It is critical to investigate and understand the problems that occur during different stages of test automation processes. This thesis is conducted as a case study and presents challenges regarding automatically analysing non-functional test results and provide improvement suggestions based on interviews at a large company in Sweden. The key contributions of this work are filling the knowledge gap in research for the performance regression test analysis automation and providing warning signs and a road map for the industry.

Contents

1

Introduction

Software testing is an important part of the software development process. It is primarily aimed to prevent faults and to guarantee a decent level of behaviour for the software product [4]. Software testing is a significant component of the software engineering studies, as the testing phase can usually take more than 40% of development efforts, and can consume even more time and effort for the systems where reliability is critical [34]. Despite its importance, software testing is also one of the least understood parts of the development process [58], and it remains as an important study area within computer science discipline [34].

Over the years, testing has become a serious challenge for software companies and these challenges are only getting more difficult as software products become more sophisticated [58]. Higher expectations of the customers have resulted in a higher demand for software products not only with more functionality, but also with more quality. To make sure the system is performing up to standards with the additional functionality, performance regression testing is required in addition to conventional functional regression testing [18]. Performance regression test runs usually take hours to days and create hundreds of performance indicators for a given system [18]. Once analysed, these indicators can give valuable information about how the system is performing in real life settings and if it is going to meet customers' expectations.

Analysing performance indicators can be difficult for companies developing complex products. Manual detection of performance regression is not efficient and error-prone due to the large volume of data that is analyzed and the limited knowledge of testers about the tested system [18]. Nowadays, while some companies rely on manual, and therefore time consuming and error-prone, approaches for analyzing performance regression tests [18], others try to automate this analysis process in order to save time and shift their employees into more critical areas where they can produce more value to the company. Automation of this analysis process is challenging, but it is usually the only choice for some companies, as the alternative can be impossible. Also, in a continuous integration setting, which most of the organizations either adopted or trying to adopt, testing should be run continuously [13]. Due to the limited infrastructure and available data, performance regression testing, in contrast to functional regression testing, lacked research interest from the academia and only few studies have been conducted concerning about this research area [18].

In order to address this gap in literature, a case study was conducted at an international company that is based in Sweden. The goals of this research are to investigate the challenges which occur during performance regression test analysis automation and suggest possible improvements for these challenges. The company decided to use a tool called Verdict System to help with the automation process and this tool will also be one of the focal points to investigate within this thesis study. During the research different data collection methods were used; static and dynamic analysis of the system, interviews, workshops and document studies. Collected data is then analysed to understand the challenges surrounding the problem domain in both the case company and other companies and offer improvements over the current systems that are in use.

The thesis is structured as follows: the next chapter presents the background and the related work in the field. Chapter 3 describes the case company and Chapter 4 explains the research methodology. The results of this study are presented in the Chapter 5. Chapter 6 presents the discussions and finally, Chapter 7 covers the conclusion and future work.

2

Background

This chapter presents relevant background information about the study, as well as related work done in the area.

2.1 Background

2.1.1 Software Testing

While software testing is the process of execution of a program with the intent of finding errors [52], it can also involve the process aimed at assess the capability of a software system and determine that the system meets the required results [24]. Software testing is a crucial step to ensure a certain level of performance and the quality of the software system. Testing is an important part of the software development and more than half of the development time is spent in testing [43].

Different types of errors can occur throughout the software systems, such as specification errors, design errors and statement errors. In order to find and address errors of all these types, there are several software testing techniques that can be used. Among these techniques are black-box testing, white-box testing, grey-box testing as well as functional and non-functional testing [52]. Within this thesis work, focal point of attention will be non-functional testing.

2.1.2 Regression Testing

According to Ghiduk, Girgis, and Abd-Elkawy [20] regression testing is the practice of validating the changed software to assure that the modified parts perform as intended and the remaining parts have not been adversely affected by the changes.

Software is modified for several reasons, including bug fixing, functionality enhancement and configuration changes. These modifications are made both during development and after deployment [20]. The purpose of regression testing is to ensure that changes such as those mentioned above have not introduced new software bugs, called regressions, in existing functional and non-functional areas of a system. This type of software testing is usually performed by re-running all the previous test cases, which is one of the most expensive and time-consuming activities that increases by size and complexity of software [30]. Brooks [7] has stated that, due to new bugs that are introduced to the system, program maintenance requires extensive system testing. In theory, every test case has to be run again after each fix to make sure the system has not been damaged by the changes. In practice this type of regression testing can only approximate to the theory as it becomes very costly to execute every test after each new change in the system. [7] Because regression testing is expensive, different techniques have been studied to reduce its cost. One such technique is called test cases reduction, which permanently eliminates test cases from the test suite. Another one is test cases prioritization, which orders the test cases by certain measures. A third one is test cases selection, which seeks to select test cases that are relevant to some set of recent changes [30].

2.1.3 Non-functional Testing

Non-functional testing is the testing of a system for its non-functional requirements, which means testing how a system operates. Network bandwidth requirements, CPU usage, available disk space, and memory usage can be considered among most typical resources that need to be evaluated [53]. The goal of this type of testing is usually performance bottleneck identification and performance comparison and evaluation [43]. As a result of the overlap in scope between different non-functional requirements, the names of many non-functional tests are usually used interchangeably. For instance, software performance is a wide-ranging term that includes a lot of specific requirements such as scalability and reliability. Non-functional testing includes, for example: usability testing, security testing and recovery testing [37].

2.1.4 Test Automation

As reported by Ieshin, Gerenko, and Dmitriev [26], test automation is using software to control the execution of tests, test result analysis, test set ups and other test control and reporting functions. A few software testing tasks, for example extensive low-level interface regression testing, might be burdensome and time consuming to perform manually. In addition, some classes of defects might not always be effective to find by using a manual approach. Test automation can be considered to perform these types of testing effectively. Automated tests that have been implemented are able to be executed rapidly and repeatedly. Usually, automating regression testing of software products that have to be maintained for a long time, will result in cost- and resource-savings. Polo et al. [44] have stated that testing takes up between 30 and 60 percent of all life-cycle cost, depending on critically and complexity of the product. Therefore, it is important to control and reduce test costs. This in combination with the fact that testing is vital and cannot simply be ignored, test automation is essential [44]. Also, test automation plays a key role together with the continuous integration practice when it comes to achieve frequent delivery of working software as well as continuous attention to good design and quality [26].

2.1.5 Performance Testing and Performance Analysis

Software performance testing is about the assessment of how the system can be expected to perform, usually from the perspective of the user [57]. By applying performance testing, information about the system's performance can be extracted. These values can later be compared to performance requirements of the users in order to understand if the system meets the expectations.

Even though the performance testing is an important aspect for many large industrial projects, there is a weakness in published work describing approaches to software performance testing [57]. There has not been any significant advance on performance testing and the tools available for software testing has been also limited [12]. In many cases, performance degradations and difficulties in handling required system throughput are the major source of problems, as opposed to system crashes [57]. Often, these software systems have gone through extensive functionality testing but lacked testing about assessing the expected performance [57]. Therefore, it can be said that in order to ensure the required performance level for a software system, software systems have to undergo extensive performance testing.

On the other hand, the goal of performance analysis is to evaluate the quantitative behaviour of a system by extensively analyzing its structure and behaviour, from design to code [8]. It involves comprehensive investigation on quantitative behaviour for each different aspect of the software system. Depending on the phase of the development where it is applied, performance analysis has to target different goals [8].

Traditional software development methods usually focus on software correctness, and only act on non-functional problems in the later stages of development process [8]. Fixing non-functional problems can require considerable changes at any stage of software lifecycle [8], but this particular development style frequently brings large-sized projects to failure [22]. Non-functional validation consists of checking if the system meets non-functional requirements at any stage of the software lifecycle, through analysis of the produced artifacts [8].

Since analysis and testing activities are closely associated, they are often confused with each other [36]. A software test analysis is an activity that produces information about the system under test [36]. A test case on the other hand has a pass or a fail result based on this information [36]. For example, in case of a web browser, a certain webpage address has been entered into the addressed bar. After the enter key has been pressed it took 2 seconds to display the webpage on the browser. These activities are analysis. A test case, for example, can contain two fields; the address (test case input) and 3 seconds (expected time to load the webpage). If the browser takes more than 3 seconds to load, the test case would be a fail result, otherwise it will be a pass result.

2.1.6 Test Oracles

A test oracle checks whether the result of executing a program using a test is right or not. The construction of an oracle can be made through plenty of techniques, including monitoring user-defined assertions when test execution is running, manually specifying expected outputs for every test and confirming if the outputs match those generated by some reference implementation, for instance, an executable model [55].

One of the well-known challenges in software testing is called the "test oracle problem". It is about differentiating the desired, correct behaviour from potentially incorrect behaviour of the System Under Test (SUT). Through test oracle automation, this challenge will consume less time to perform and therefore result in greater overall test automation. Without test oracle automation, a bottleneck will occur where the human has to manually decide if observed behaviour is correct. The research in the area of test oracles is mainly about techniques for oracle automation, including modelling, specifications, metamorphic testing and contract-driven development. In contrast to these techniques, the human is the final source of test oracle information. The reason behind this is that the human may be aware of informal specifications, norms, expectations and domain specific information that bring informal oracle guidance. Increasing benefit and minimizing cost are two challenges that are associated with all kinds of test oracles, including the human [3].

According to Barr et al. [3], there are four major approaches to the solution of the test oracle problem. Test oracles can be specified, derived, built from implicit information or there is no automated oracle available, but it is still possible to reduce the human effort. With specified test oracles, a specification language is the notation for defining the test oracle that judges the behaviour of SUT according to the formal specification [3]. Solutions then depend on the degree of abstraction and on the implementation of the system under test. Gaudel [19] stated that a successful test driver is not guaranteed by the existence of a formal specification. A derived test oracle uses different artefacts such as documentation or system executions to judge a system's behaviour. Derived test oracles are usually chosen when specified test oracles are unavailable [3]. An implicit test oracle relies on general knowledge that is true in almost every case such as "buffer overflows and segmentation faults are nearly always errors" to assess SUT's behaviour and since it is not domain specific it can be applied to nearly all programs [3]. Since implicit test oracle only evaluates limited aspect of the SUT it can only be considered as a partial solution. The first three solutions require a kind of artifact which serves as the core for a full or a partial oracle. As these artifacts are seldom found in industrial settings either due to difficulty of applicability or lack of coverage, out of these four solutions, the last one, human oracle is the most commonly used one in the industry. With human oracles, since there is no artifact available to verify the correct behaviour of the SUT, the responsibility belongs to human testers. At this point, software engineering tries to investigate possible ways of reducing the work required and automate the process as much as it is possible.

2.1.7 Verdicts and Verdict System

A verdict is one of the three possible statements, ”pass”, ”fail” or ”inconclusive”, that is given as a result of how the SUT performs according to standards set in a test case. [27] The verdict pass indicates that the observed behaviour matches the specification and the test objective has been reached. The verdict fail means that the observed behaviour is a failure and is not consistent with the specification. The verdict inconclusive is used when no failures have been observed, but the test objective has not been reached. For instance, if a test case has to set up a connection over an unstable channel in order to test a specific application and it fails to set up a connection then this does not represent a failure but also does not test the application [25].

A special form of a test oracle is the verdict machine which is the object of our case study. A verdict machinery determines if the results of executing a program using a test case has been passed or not or results are inconclusive. The verdict system can be considered as a combination of a human test oracle and a script which compares the test results with the expected output. It was stated by Barr et al. [3] that, in many cases, as it was the same situation with the case company, there was not a specified, derived or implicit oracle available that helped with the automation, so verification of the correct behavior of SUT was a human responsibility. A verdict system was created and later developed by the department in order to reduce the overload on testers, which succeeded initially. However, as time passed and verdict system has become a central tool, which testers within the department relied upon, several new problems arose. Witnessing the verdict system in its initial steps provided the opportunity to observe the challenges of test analysis and test result validation automation in real life settings through this case study.

3

Case Company

This chapter narrates the case company and case department where the study was done. First, a general information about the company is presented. A detailed outlook to the case department is shown and finally the chapter ends with a description of the tools within the domain.

3.1 Case Company Description

The case company is a multinational provider of communications technology and services. The company offers services, software and infrastructure in information and communications technology for telecom operators and other industries. This infrastructure includes traditional telecommunications, Internet Protocol (IP) networking equipment, mobile and fixed broadband, operations and business support solutions, cable TV and IPTV as well as video systems and an extensive services operation. According to the annual report for 2014, the company's current position is number one in mobile infrastructure, OSS and BSS, telecom services and TV platforms. The company has also the number one position in the LTE market share in the world's 100 largest cities. Other company facts include 37,000 patents, 180 countries with customers and 118,000 employees.

3.2 Evolved Packet Gateway

Number of smartphones and smart mobile devices is rapidly increasing globally and it is expected the be over 3.7 billion by 2017 [14]. These devices enabled users to connect to internet everywhere within a network connection and this recent phenomenon resulted with customers demanding high quality broadband. This demand for a network service that provides high availability, high speed and low latency resulted in stricter non-functional requirements, low number of errors and consistency from the product at the case company.

In order to meet high expectations of the customer, case company has set customer requirements accordingly. As a result of high demands, understanding how the product is performing based on its scalability, capacity and similar criteria is critical to the success of the department and therefore, at the EPG department, all products, services and new functionalities are tested thoroughly.

The Evolved Packet Gateway (EPG) is a component of Evolved Packet Core, which provides a solid foundation for delivering mobile data services [15], and it is developed to be a critical part of LTE networks. EPG as a product can be valuable for service providers, as they can use it as a gateway between their mobile packet core network and the Internet [15].

EPG system at the case company is a complex product that achieves to offer high scalability, performance and availability to the service providers. In order to keep the high standards that the customers expect with the product, each new build has to periodically go through extensive testing phases. These daily, weekly and monthly tests cover certain areas and guarantee that the product is performing up to certain standards. Performance degradations with a certain extension to the product can hinder the level of quality and result in customer loss for the company. To prevent losing revenue, test phases have to inform the department in case of lower efficiency, higher upkeep and other potential inadequacies.

3.3 Verdict System at the Case Company

The verdict system used at the Evolved Packet Gateway at the case company is a computer script to automatically analyzes test results. It can analyze a large number of test results within seconds and its purpose is to accelerate the analysis in order to cover more testing with the same amount of testers. The system is written in the Python programming language and has a simple loosely coupled design which makes it easy to extend the automatic analysis, see Figure 3.1.

Given a set of checkpoints, the verdict system analyzes a test case and for each checkpoint it returns a result of either pass or fail. If all checkpoints have passed, then the corresponding test case has passed too. A test case is a class which has methods to read log files containing test results, in the form of Key Performance Indicators (KPIs), also called performance counters, generated by the test execution. There are hundreds of different KPIs that measure different properties of the SUT, for example CPU and memory utilization as well as throughput. A checkpoint is an interface which has method signatures that have to be implemented with specific test definitions. Various checkpoints exist which test various quality attributes, including capacity, robustness and stability. Checkpoints are also of different types regarding how they are constructed. One type is when a checkpoint compares its KPIs with the corresponding pre-defined threshold values and another type is when a checkpoint compares its KPIs with the KPIs from previous test executions. The former type determines whether the properties of the SUT fulfill the specified non-functional requirements, whereas the latter type aims to search for performance degradations.

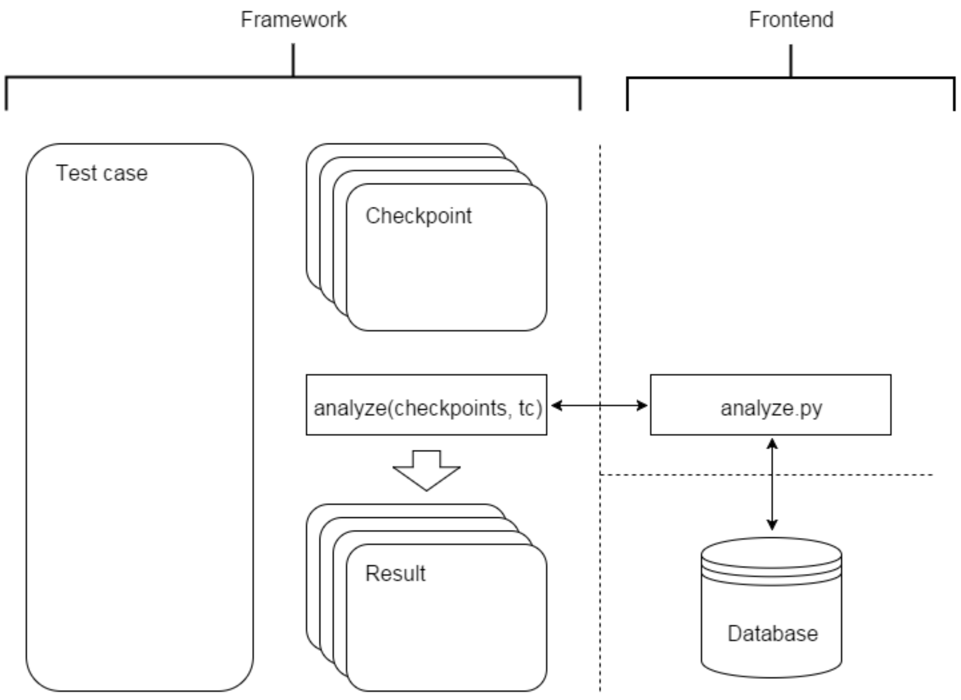
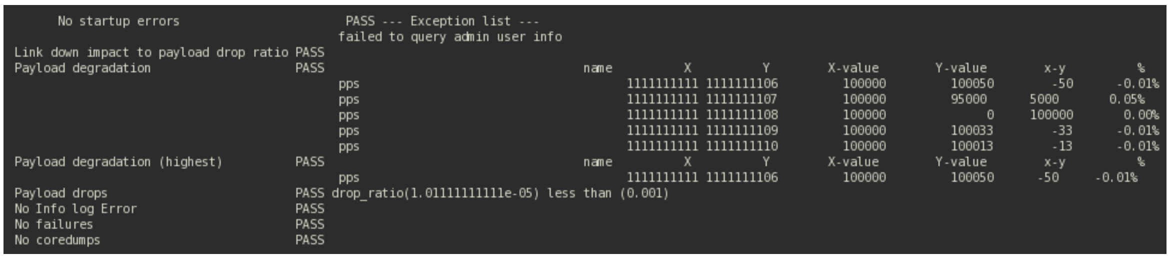


Figure 3.1: Architecture of the verdict system used at the EPG department

A typical flow of the process is as follows: First, analyze function takes two arguments as inputs; a test case object and a list of checkpoints. The checkpoints are created by system testers and contain information about which KPIs within the test cases have to be analyzed (prerequisites) and what values are required to get a PASS verdict. The function then decides if checkpoints are valid and performing up to standards based on the test case attributes. Finally, it prints the results on the screen and stores them in the database.

An example of how a capacity test case is handled through the verdict system has been given below. There are three main categories for capacity test cases; payload, PDP (Packet Data Protocol) and signalling. A payload test case will be the basis of this example. Even though these test runs are usually done automatically it is also possible to configure and run them manually. For this example latter approach has been chosen to demonstrate a clear and explicit process. An example of a final verdict has been shown in Figure 3.2.

As the first step the product which the test is going to be run on is selected. The product in this case is the EPG component that provides GSM, LTE and WCDMA connectivity between multiple users and Packet Data Networks (PDN) such as Internet and LAN. Secondly, a capacity payload test case which the product is going to be evaluated and graded upon has to be chosen. The payload test cases usually include requirements about the maximum rate of successful message delivery over the system under test. This rate is also called throughput. The test cases hold information about



```
      No startup errors                    PASS --- Exception list ---
                                          failed to query admin user info
 Link down impact to payload drop ratio PASS
 Payload degradation                     PASS                                  name          X          Y         X-value       Y-value       x-y          %
                                          pps                                        1111111111 1111111106        100000        100050        -50       -0.01%
                                          pps                                        1111111111 1111111107        100000         95000      5000        0.05%
                                          pps                                        1111111111 1111111108        100000             0    100000        0.00%
                                          pps                                        1111111111 1111111109        100000        100033        -33       -0.01%
                                          pps                                        1111111111 1111111110        100000        100013        -13       -0.01%
 Payload degradation (highest)           PASS                                  name          X          Y         X-value       Y-value       x-y          %
                                          pps                                        1111111111 1111111106        100000        100050       -50     -0.01%
 Payload drops                           PASS drop_ratio(1.01111111111e-05) less than (0.001)
 No Info log Error                       PASS
 No failures                             PASS
 No coredumps                            PASS
```

Figure 3.2: Log file of a successful final verdict

what needs to be tested, how it has to be tested and in which environment it has to be tested on. If a tester tries to run a test case on an incompatible product, the test would not be executed. After correctly choosing the test case and the node, the test is finally executed. The test runs produce a great quantity of data for testers to investigate which can create a bottleneck within the testing process. This is where the verdict system helps the department by providing a summary of the most critical pieces of information produced by the test run. It takes the requirements stated in the test case definitions and scans through all available logs to get the necessary information and presents its findings to the tester. In a typical payload test case, requirements are usually about having a minimum level of throughput, having a limited degradation in throughput and not having core dumps or unexpected error messages. The accepted degradation levels are set by object responsibles in collaboration with and tested accordingly, but in some cases, such as test runs with core dumps and environment problems which will be covered in Results section, the verdict system becomes unreliable to give a verdict on degradation. If these requirements are met, which means that the system was able to produce a high level of throughput that is above the requirements and not below of a certain level of the previous test runs while not having any unexpected errors, the verdict system prints out in a log file that each requirement is passed. In case a requirement is not met, the verdict system still shows that while other requirements are met, due to system not performing up to standards for one requirement, that test case is failed. The verdict system goes through the logs and looks for keywords, such as ”ERRORS” and ”Value”. Once these data pieces are compared to previously stated requirements the verdict system makes a verdict and prints out the result as a log file. It is then tester’s responsibility to make sense of the verdict and to decide if the test is really passed or not.

4

Method

This chapter describes the purpose and the methodology of the conducted research. First, the research questions are presented. Then, research method is explained, followed by the descriptions of data collection and data analysis approaches.

4.1 Research Questions

The purpose of this study is to understand the challenges surrounding automatically analysing non-functional regression test results and propose possible solutions to these problems. In order to achieve these goals, the following research questions are addressed:

- **RQ1**. What are the major challenges of automatically analysing non-functional regression test results?
- **RQ2**. How can the automation of non-functional regression test analysis be improved with regard to accuracy and usability?

4.2 Research Method

A case study approach was adopted as the research method for this study. The necessity of examining challenges that occur during the automation of non-functional regression test analysis and exploring possible solutions in the real-world setting was the main reason for choosing this approach over other methodologies. A case study allows researchers to study the phenomena in its natural environment, thus understand how it interacts with the context [48]. Moreover, it is well suited for software engineering research, as the phenomena is difficult to study in isolation [48].

According to classification by Robson [45], there are four kinds of purposes for research; exploratory, descriptive, explanatory and improving. The purpose of this study is a combination of exploratory and explanatory, as the aim is to discover the challenges in this field in industry, describe them and offer possible solution suggestions to these issues. Qualitative research method was chosen because it is exploratory and particularly useful when the researchers do not know the important variables to examine [9]. Furthermore, if a phenomenon needs to be understood because little research has been done on it, which was the case with the subject of this study, it will be beneficial to select a qualitative research method [9].

4.3 Data Collection

It is stated by Sapsford and Jupp [49] that measured data has to be in correlation with the arguments that are made in the thesis and any conclusion that is drawn at the end of the study. In order to answer research questions of a study, the data has to be collected first. There are different methods of collecting data and they have their advantages and disadvantages. To limit these negative aspects of each data collection method, data triangulation was used for this study. Triangulation method, which is the use of at least two methods to address the same problem, can be used to ensure that the most comprehensive method is used to solve the research problem [41]. Triangulation method can also be helpful for researchers to improve the accuracy of their perception of a problem by collecting different kinds of data about the same research domain [29]. Considering all of these points, it was decided to use a combination of multiple data collection methods for this thesis; interviews, workshops, document studies and analysis of the system, as all of them would be beneficial in different aspects.

Understanding what the precise nature of the encountered issue at the case company was the first goal of the study. A better validation system for test results with better accuracy and more friendly user interaction were stated as the expectations by the case company, but it was necessary to conduct further investigation to understand the root causes for these goals. In order to achieve these initial goals, several interviews were made within the case company, as well as other companies. Data gathered was critical to understand the underlying causes of the difficulties faced within the domain and shape future suggestions that were supposed to be presented in the end to the case company. Interviews with other departments and companies presented valuable data to discover how to make this study be applicable to different companies and cases, hence generalizing it.

During the course of the study, the researchers have attended multiple workshops that were held by the case company, several interviews have been conducted, the system was analysed and the online and written documentation was scanned through. These four methods were not only provided insights about the research questions, but they also helped the researchers to gather enough technical knowledge that they may lack about the tools used within the case company and industry.

4.3.1 Document Study

Documentation that was available at the case department's online archives was studied as one of the first steps to improve the knowledge about the domain. Studying documentation can serve as an introduction to the software and the team [33], and it was used as an initial step to understand the domain and the case department. The information which was found as a result of these inquiries is carefully studied and analysed. Apart from the department's own documentation, an online enquiry about test result validation systems was made. The lack of consensus about the terminology for the subject was first discovered during the application of this method and several topics with different keywords are used to find more information about the subject.

As a result, all of the documents, academic papers, books and workshop summaries about test oracles, regression testing, test validation and non-functional testing that were studied within this method helped lowering the gap between the knowledge available about the subject and the current state of information about the systems the researchers had. It was also noted that the lack of agreement about the certain keywords had to be considered during the course of the thesis work.

4.3.2 Interviews

Interviews were the most time consuming part during the early stages of the study and they served as an introductory stage to both the domain and the case department.

Interviews are an effective way of exploring how participants experience [51], and it is because of this reason they are used at the beginning of the study. The lack of knowledge about the domain and obligation to understand why and how the problem is affecting the employees, as well as the business itself, made it critical to resort gathering employees' opinions about the procedures about the way of working and other technical details. Even though interviews are not good for eliciting data, they can be used for gathering opinions, goals and procedures [50], and in order to analyze the origination of this problem and its effects on the business flow, interviews were held initially, and before constructing a prototype or a solution for it.

Semi-structured interviews were selected as the interview method to follow. Since semi-structured interviews allow for improvisation and exploration [48], it was possible to have a core set of questions that enabled interviewees give insight about their individual perspectives. The questions are constructed in a way to gain most about the possible answers to research questions.

A typical interview takes around 45-60 minutes and begins with an introduction of the subject and researchers to the interviewee. After this introduction sound recording device is turned on and it begins to record the conversations. The researchers can take notes during the interview, or take pictures if interviewee decides to explain a phenomenon with the help of the board. After the interview, a short brief has been made and interviewee is thanked for the input. After the interview is completed, the transcript has to be written and later analysed. This analysis will eventually result in findings. Apart from one interview, which was done in a premise of another company, all interviews are conducted in case company's campus. Out of 11 interviews that were conducted in case company's campus, 3 of them were done remotely and Microsoft Office Lync tool was used as the communication device.

The problem, goals and expectations had to be understood clearly at the beginning of the study. Several meetings were held together with the test architects, who were also the supervisors for the thesis work, to make sure that a common ground has been found. Discussions with the test architects also included topics such as;

- How does architecture of the verdict system work?
- What are the most complained issues about verdict system?
- Who should be interviewed at the department to get the most knowledge out of the verdict system?
- Who should be interviewed at the other departments that can provide an outsider look towards the department, but still have relevant experience with the subject?
- Who should be interviewed at other companies which can use similar tools with complexity such as EPG?

During the meetings with the test architects, it became evident that while the goals of the project were clear, understanding the context of the problem itself was not. The initial goals for the study as they were stated in the job description can be summarized as further improving the verdict system with respect to its accuracy and usability. However, the underlying aspects of the problem domain were challenging to grasp, and even when they are understood, they were difficult to explain in a formal context. In order to overcome this deficit several interviews with subjects both from the case department and outside were conducted. As a result of this study the architecture of the verdict system and how it works were understood.

In total, there were 12 interviews that had been held with the people who had various levels of experience, knowledge and background. The interview subjects can be divided into 3 major groups according to their places of work;

- Case department at the case company
- Other departments at the case company
- Other companies

This distinction benefitted the study in differentiating how each individual person from each group can provide particular collection of information due to their diversity and proximity to the problem domain.

The reason why the majority of the interviews were at the case company was that this study aims to, especially during the initial phases, understand the challenges regarding the verdict system at the case company and then, later, improve it combining this

Company	Department	Interview number	Title	Experience
Case Company	Case Department	1	Function Tester	10+
Case Company	Case Department	1	Systems developer	10+
Case Company	Case Department	1	Software designer	10+
Case Company	Case Department	2	Systems manager	5
Case Company	Case Department	3	Systems Tester	7
Case Company	Case Department	3	Software developer	10+
Case Company	Case Department	3	Verification engineer	7
Case Company	Case Department	4	Consultant	10+
Case Company	Case Department	5	System test specialist	10+
Case Company	Case Department	6	Line manager	10+
Case Company	Other Departments	7	Line manager	10+
Case Company	Other Departments	7	Developer	10+
Case Company	Other Departments	7	Developer	8
Case Company	Other Departments	8	I&V discipline driver	10+
Other Company 1		9	Quality assurance engineer	9
Other Company 2		10	Chief lead for continuous integration and test governance	10+
Other Company 3		11	Technical expert	7
Other Company 4		12	System engineer	10+

Table 4.1: Information about interview subjects

information with the additional data gathered from other sources. Table 1 contains the titles of all the people that have been interviewed, the names of the departments they belong to and the names of the companies where they are employed.

First group, case department at case company had the most information about the system and its problems. Employees were more experienced with the system at hand, familiar with the expectations and results. Having a day-to-day involvement with the system can be beneficial and any kind of struggles can be extorted during the interviews. Since people in this group knew more about the problems and had familiarity with the researchers, a consensus on the terminology can be reached and introduction phase of the interview and explanation of the problem can be kept to the minimum. Due to familiarity with the verdict system and expected improvements, information about what aspects of the system was underperforming or up to standards could have been extracted from employees within this group. Any kind of comments about expectations, possible improvements and plans for the future, as well as more information about how the system works and how it actually should work were also gathered from this group's subjects.

Interviews with employees from other companies can provide distinct insights and outsider point-of-view. It was not known whether the same problem was encountered at the other companies, and if so the reasoning or prevention behind this situation can be studied. There are several questions that can be answered only by people outside of case company, such as if there are any company specific terminology being used for the same problem.

The interviews with the people from the case company who are outside of the case department can provide the characteristics of previous two interview groups. While these employees are not from the same department, they can still provide information that is company specific, and at the same time they can contribute knowledge from a different perspective.

4.3.3 Static and Dynamic Analysis of the System

The case system, as well as the source code, at the case department was also made available to use, investigate and test upon. The so-called verdict system is an application that is used by the testers to reduce the work significantly and help them analyse and find the cause of certain problems. The system was first created in 2013, as an idea by another thesis work, and then programmed into what it is today. The application is programmed in Python programming language and information about how the system should be used can be found online at the portal of the department. Since the application was an offline tool, meaning using it would not in any way affect the department, it was safe to use it with different configurations.

The system takes a log file of the test results as the input, and according to expected requirements, creates a verdict that can reduce the work into seconds that would otherwise can take hours done manually. The researchers were able to get a first-hand interaction with the application and run it with several different inputs and cases. This interaction gave a different perspective to the researchers and helped them to understand the difficulties users can face while using the program.

4.3.4 Workshops

During the course of the study, researchers have attended several workshops that were being held by the case company. The workshops took 2 to 4 hours and were attended by both researchers at the case company campuses. Each workshop covered different grounds which provided the opportunity to understand distinct aspects of the department which the thesis work was being held, the way of working within the domain, technical details and about the different methodologies which could have been applied to this study. Within the workshops, the researchers were able to be apart of a group with other employees and interact with more experienced members of the group. These events helped to gain from the experience and increase the knowledge about the domain, as well as introducing the researchers to the people who were going to be participants in the study.

4.4 Data Analysis

Data analysis was the next step after the data collection. Collected data through some operations was organized, the meaning was extracted and conclusions were drawn, and as the final step, theories can be written which describes this data [54]. In the following chapter, the methods which were followed to make sense of the data are described and explained.

4.4.1 Interviews

Data analysis for the interviews was done according to the guidelines provided by Romano Jr et al. [46]. Three major parts of this approach are; elicitation, reduction and visualization. Furthermore, reduction phase consists of three subsections; selection, coding and clustering.

Elicitation

Elicitation phase consists of collecting or recording the interactions in words [46]. In this part of the analysis, researchers started collecting data and documenting it down in a systemized fashion. As a result of this step, all the transcripts of interview sessions were written down and collected in a single storage unit. Notes, pictures and other types of information taken during the interview were also mapped to the regarding interview session in the document. Also, it was in this step the researchers began familiar with the available data and notice certain patterns. Initial thoughts and ideas were recorded.

Reduction

Reduction is the phase that is responsible for selecting, focusing, simplifying, abstracting and transforming the raw data collected into something useful [46]. This phase consists of three subsections, which are described below.

Selection section of reduction phase is about choosing categories and creating schemes and distinct word lists [46]. In this section, initial notes and records were used to create categories which available data can be grouped within. Collected data during elicitation phase later broke down into these separate categories based on their content. This sub-categorization resulted in better understanding of the problem, how the problem domain can be improved with respect to its accuracy and usability and how a practical solution has to be constructed in order to avoid these problems in the future.

Coding is the categorization of data into conceptual categories [47]. Codes are labels or descriptive texts that are used for differentiating units of meaning to information compiled during the research [38].

As the first step for the coding phase, all of the descriptive information about each of the interview transcript is read and reviewed. Then, according to their similarities, these notes and comments are categorized and grouped together. This process, along with code creation and regrouping is repeated until a coherency was found between the codes and notes. In order to reduce the risk of inaccuracy, both researchers took part at some points together and in others took turns during the process.

Clustering involves choosing which data to code and how to code them [46]. As it was the case with the previous chapters, this step was completed by the collaboration of both researchers. Each note and object within higher proximity and affinity was put into same cluster and as a result of this process, a higher similarity is found between the notes within the same clusters as opposed to others. The result of clustering is used to generate graphical and textual visualization in the next phase.

Visualization

Visualization phase consists of creating organized, compressed assemblies of the coded notes that allow conclusions to be made [46]. In this phase, researchers have illustrated similarities and shared proximities with patterns that are based on the codes that were created in previous phases. In the end, as a result of this phase, conclusions and results can be seen and found.

4.4.2 Documents Study, Static and Dynamic Analysis of the System and Workshops

Data analysis for documents study, system analysis and workshops were done using qualitative content analysis. In qualitative content analysis, data are grouped with the help of categories that are generated inductively, and in most cases applied to the data through close reading [40].

After the data was collected from studying documents that were specific to case domain, data analysis part was soon followed. Data analysis phase improved the knowledge acquired about the subject, lessened the technical knowledge between the current ability of the researchers and what was necessary to know to understand the codes, architecture and similar aspects of the systems used. This process was repeated each time pieces of information are gathered from the workshops and system analysis and all data that was gathered are analysed in the same manner. This process helped researchers to grasp about the relative research areas and learn what the focus of the study should be about.

The method that was used for analysing these data was based on the inductive category development process which was explained by Mayring [35]. According to Mayring [35], the procedure begins with formulating a criterion of definition that is based on theoretical background and research question, followed by going through the collected material and provisional categories are created and later deduced. This process is repeated and categories are revised and reviewed until main categories are found and a reliability has been found.

Categorization began as soon as information was retrieved and notes and memos are collected and written down. Common keywords and parts that had insights about the research questions were generated and put into categories according to their similarities. After much revision and reviews, categories are changed and contents are changed. This process continued until all of the materials are analysed and used within categories and as the final step, a general review is done to ensure the reliability of the analysis.

5

Results

This chapter narrates the results and findings of the study. The results compromises the analyzed data gathered from the various data collection methods applied throughout the study. First, the challenges regarding automation of regression test analysis is presented. Then, improvement suggestions are showed.

5.1 RQ1 - Challenges regarding automation of performance regression test analysis

This section covers the research question related to challenges regarding automation of performance regression test analysis. The identified challenges are presented in the form of a thematic map in figure 5.1.

5.1.1 Infrastructure

The infrastructure set at the case company has created some challenges for the automated analysis process. These challenges are listed below.

Continuous integration. The comments about the effect of continuous integration on the analysis process were diverse. Two of the participants have explained that continuous integration makes analysis complicated and difficult. A systems manager has stated:

> *"It's harder basically, if there's a capacity degradation it is hard to see where that capacity problem is introduced."* — Systems Manager

Another interview subject has said that;

> *"The problem is that this daily scope is not very well tuned into continuous integration ways of working. In the continuous integration view it says that*

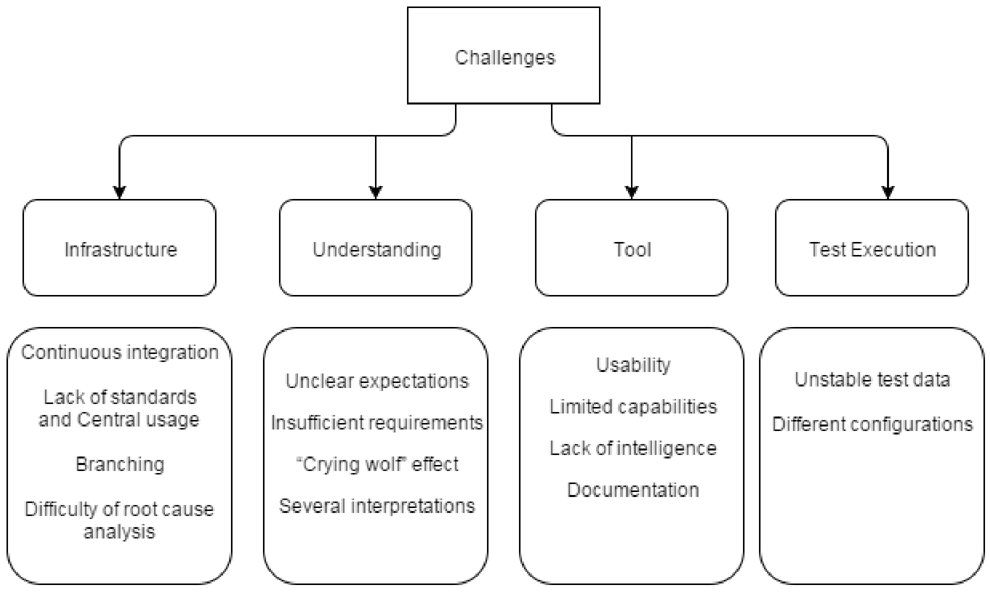


Figure 5.1: Thematic map of challenges

> *you should not do your troubleshooting when it is delivered, you should do your troubleshooting before or after it is backed out from the delivery. So, in that sense all the test cases are to just make sure that you have a working build and you can push in as much testing as possible. This is what we want to guarantee with overworking build. And if it fails, then we do not really care to look at why it fails, we merely want to know which change caused this build to fail and then we back out that delivery and we go back to the state we had before of working build and then we do the troubleshooting outside that. But the daily scope is not very well tuned to that, because in one day we can have like a hundred commits to the master branch. You might have two, three or four new failures like new core dumps, decreased capabilities, some kind of new alarms or whatever. It is very hard to find the faulty commit in that consequence."* — Product Owner

This means that there can be clashes between the work that is actually done and how it should be done using the continuous integration. Another positive stance the participants had was that using continuous integration resulted in informing users of the problems earlier than before. Line manager has stated that;

> *"Continuous integration is good since bugs can be found earlier and easier because of their shorter trace, which leads to that less time and resources have to be spent on addressing the root causes of the bugs."* — Line Manager

By putting pressure on faster feedback cycles, continuous integration helped verdict system informing the users of the system in a shorter period of time.

Although there were some positive opinions on how continuous integration helped discovering faults within the SUT in a shorter time span compared to traditional development methods, two of the participants have stated that continuous integration made it difficult to pinpoint where a fault is introduced to the system. Due to continuous integration and long run times of the system tests, it is possible to have tens or at some cases hundreds of commits between two test runs. In case of a performance degradation which results in failed test cases, it is not clear which commit or group of commits actually caused the problem. Therefore, it becomes testers responsibility to dig into each commit and try to figure out the faulty one that is causing degradation. It is impossible to run system tests after each commit, as it was stated by Duvall et al. [13], it might be catastrophic to even try it. Another participant has stated that due to this reason, their daily scope of was not tuned with the continuous integration style. The complexity of the system means that there is extensive analysis time for test cases, the need for following-up errors, take logs and write problem reports which causes problems with continuous integration way of working. As a result, problems such as this one cause delays and additional workload while the case department was trying to automate performance regression test analysis process.

Lack of standards and central usage. One of the learnability issues, which corresponds to lack of matching between the system and the real world, is that there are naming inconsistencies within the tool that is used at the case company. Two participants have pointed out that they have had problems with the naming, as names of checkpoints, errors and other properties can be inconsistent and do not following any kind of documented standards. They also claimed that some of the object names do not have any explanation of what the checkpoint is about and most of the time they are just abbreviations. A product owner has stated that:

> *"Since on the top-level you only get this very cryptic checkpoint name, so it is just letters and numbers."* — Product Owner

In the department during everyday work, naming inconsistencies lowers the understandability and makes it difficult for users, especially the new ones, to learn the system.

Department manager complained about the lack of a central usage about the tool. It was pointed out that the tool had documentation, GUI and output screens spread across several different platforms and this situation caused inconsistency and made it difficult to use. It was also stated that:

> *"It would be better to have a centralised place for all the non-functional requirements in the verdict system, instead of having multiple of the requirements outspread in different checkpoints. This improvement will increase the performance, usability and maintainability of the system. In order to do this improvement the architecture of the system has to be changed."* — Department Manager

In order to improve the quality of the system and make it easier to use, a new architecture with more central based approach has to be created according to department manager.

Lack of consistency and standard of checkpoint names used within the tool were also mentioned during the interviews by two participants. A test system specialist has mentioned that:

> *"It is better to have checkpoints and they belong to everybody and we can just for test pick any of them, except that nobody is using it like that actually."*
> — Test System Specialist

This points out the finding that there is a disagreement on how to create checkpoints, to whom they belong to and what are the privileges each user of the system has on the checkpoints.

Interviewees' remarks about naming inconsistencies seem to suggest that there is a mismatch between the real world entities and their representation in the system. Use of code names and abbreviations made it difficult for testers to evaluate the problem initially and required them to examine other available material such as logs to understand the system's output. Tristem [56] has also stated that lack of similar terms to natural language in a system can cause usability issues and stated it as a problem. Furthermore, software product's capability to adhere to standards was one of the complained about aspects according to results of this study. Users of the system have stated that they were having problems due to inconsistencies with the usability of the tools and lack of standards. Another point of difficulty for the new users of the tool was lack of standards with the checkpoints. Testers became unsure if a certain checkpoint belong to their group and they have clearance to use them. Miller [39] states that similar things should look and act similarly, as different things should be visibly different. Lack of this distinction prevents the new testers to grasp some of the key points within the tool.

Two participants have stated that tool's lack of central user interface hinders performance regression test analysis automation process. Not having a single central user interface complicates the user-system interaction and makes it difficult to be efficient with it, especially for new users. This was also a problem for the managers, as some of the testers and developers have their own test cases and configurations to work with which could have been useful to other employees. This approach dangers the teamwork among the employees and prevents the potential of collaboration.

Branching. Branching constitutes the underlying reason for a big portion of the problems with analysis process. During the development process, developers branch out on one of the previous builds to work on their local repositories. After the coding is over, developers commit their changes back again to the central repository where performance regression tests are done. However, this way of working is prone to several issues as requirements, boundary values and acceptance levels for each build can be different from each other. A developer who branched out on a certain build is essentially affected by the performance of that particular build. Even though the changes the developer made within that code is up to standards and optimizes the performance, due to initial low quality of the code, the final version might not perform up to standards according to performance acceptance tests and thus, be labeled as failed. On the other hand, the vice versa is also problematic. If a certain developer puts a bad piece of code that affects the performance negatively, it might go unnoticed if the build that was branched out on is an exceptionally performing one. In this case, the performance of the final build will be up to standards, but it could have been even better. It has to be also noted that this issue is related to continuous integration way of working and insufficient requirements. Several developers working with a number of different branches make it difficult to automate the test analysis process, since it will be a major challenge to get the updated requirements and acceptable boundary values for performance indicators.

Another problem that arises when an extensive feature has to be implemented into the system. For example, introducing new features can decline the speed and the CPU usage of the system and therefore, does not allow the system to perform on a desired level of minimum quality as it was stated in the initial requirements. However, a certain level of decline can be tolerated and even expected, but since the tool used at the case company is just a simple comparison tool, it fails to give the expected final verdict. With the previously stated examples, the verdicts given by the automated verdict system are not the desired verdicts, even though they meet the initially stated requirements. After a manual inspection is done, a tester can see that the verdict given by the system is wrong and it is changed manually to the opposite verdict.

Difficulty of root cause analysis. Several interview subjects have many times mentioned the difficulty of finding the root cause of an error or a failed test case during test analysis. Due to several reasons, such as continuous integration, complexity of the SUT, high volume of available data, large number of developers working with different branches makes it difficult to locate where the problem is first introduced to the system. A system developer have stated that the verdict system only indicates the existence of something wrong with the system, not what it is and it is developer's responsibility to understand the cause of that problem. Some participants have also stated that such attempts might be futile. Even though the verdict system were not designed initially to locate the errors, it would take a huge amount of effort and time to integrate that functionality. One product owner have stated that:

> *"Well, if you spend 1000 hours just on trying to implement a program that find the root cause of this specific problem, then you have another problem next week."* — Product Owner

5.1.2 Understanding

The results of this study indicate that subjects and concepts about performance regression test analysis automation can be interpreted in different ways. Stakeholders have given different statements about definitions, goals and conditions. The challenges below are related to this phenomenon.

Unclear expectations. The results of this study show that interviewees had different expectations from the verdict system. While some participants were satisfied with the tool's ability to indicate errors, others were complaining about its inability to pinpoint the cause for that error. An operation product owner has stated that current tool used at the case company is valuable only if it is used in the right context:

> *"When we use it in the right way, it just raises the flag and say there is a new problem and then we do not do any problem analysis, we just try to find the faulty commit and back it out and then do the troubleshooting outside. In that situation it is a very strong and it is a very good tool."* — Product Owner

However, if something more is expected from the tool, then employees had issues:

> *"If we try to have a more comprehensive look of how the system is performing, then it is a very bad tool. When we actually find errors, then it is also a very bad tool to indicate where the error is and it does not produce any automatic trouble report, it does not pinpoint you to the error, it does not say what is the root cause or anything. It just say that we have something wrong."* — Product Owner

Another participant has stated that they would have expected the tool to filter out problems within the test data before analysing:

> *"In my ideal world, we should be able to produce a verdict system that sorts out this stuff, with the help of statistics. It doesn't have to have the intelligence that we have but it is very easy to see this problem."* — Function Tester

A verification engineer has also expected the tools to find the root cause of the problem, on top of pointing out that there is a problem:

> *"I think it will be nice to have this verdict system to analyse the test case to find the root cause why this checkpoint has failed."* — Verification Engineer

There were additional expected goals from the verdict system, apart from validation of the test cases. Three participants have stated their desire to use the tool as a warning system for possible problems that does not hold any risk in the short term, but could cause issues in the long term. Participants have stated that the indicators were there for a manual tester to see and it was possible for an automated system to pick up these signs and warn the testers about the lowered quality of the SUT. In that case the SUT would perform up to standards set by requirements, but since it performed worse with every new built, the tool would warn the testers about the possible risks.

The results of this study has shown that participants had different goals and expectations from the performance regression test analysis and its automation process. Even though some interview subjects had agreed on limited functionality should be expected from the tool that is used, others demanded more functionality and capabilities. While some testers were happy with what the tool was providing, and claimed that expectations have to be lowered, others have expected tool to be more accurate, more user friendly and on top of informing testers in case of a malfunction, describe what that malfunction is in detail and point to the root cause of the problem. All of these different expectations point out that what is demanded from this process can lead to disagreements and tools that are devised to solve this problem can not satisfy all of the stakeholders. As it was stated by Larsson and Borg [32], it is difficult to align goals within large organizations, such as the company in which this study was conducted, as each different department has their own perspective and goals to think about. Difference in satisfaction levels for the tool does not affect its performance, but it might affect the morale and incline to use the tools for employees in some of the departments.

Insufficient requirements. One of the challenges for automation of performance regression test analysis was that requirements for either passing or failing a test case were not always sufficient. Due to high number of test cases, KPIs and dynamic test environment, stated requirements could be insufficient or outdated after a while. An operational product owner has stated that in some cases with newly added functionality a need for new requirements arises:

> *"One other thing also with system test is that not everything is specified like a black and white true or false. For example, if we restart on a card and we have a requirement that says: it should be up and running within ten seconds. However, after a software delivery it could be 11,5 seconds, so, is that totally wrong or is it OK? The increased time could depend on a number of things. It could depend on one other feature increasing the load on the system. You have to ask someone, who says: OK, we can live with it and then we change the checkpoint to say 12 seconds is the new limit."* — Product Owner

Consequently, due to the intricate nature of the many-to-many relationship, it is imperative to conduct meticulous change impact analysis in order to ensure that the introduced changes does not have a negative impact on the test coverage of other requirements, as described by one system developer/technical project manager:

> *"if you have to change a requirement then you have to change a test case [linked to the requirement], and if you change a test case you have to make sure that the test coverage of other requirements [traced to that test case] is not affected"* — Software Developer/technical project manager

The complex and dynamic development processes force testers to update the requirements accordingly, but it is not always clear how to complete this action without help.

Several anecdotes were shared during the interviews and meetings with test architects about how the nature of the problem is dynamic and there is a need for dialogue to adjust the requirements. In one scenario, a developer knows that there will be a hardware upgrade next week for the SUT, which will raise the acceptable value for the memory consumption KPI. So, this developer integrates new functionality to the SUT, even though he is aware that the memory consumption would be more than what is permitted at that time. When SUT is tested, according to initial acceptable memory consumption levels the test has to be failed and thus verdict system fails the test case. However, testers are also aware of this situation and decide that verdict for this test case has to be overruled and they pass the test. Another case that has been discussed is also about the memory consumption. In this scenario, the developer commits the new code into the system and it has seen that memory consumption is in the acceptable levels, so verdict system decides that test case is passed. However, when manual analysis is made the testers saw that there is a big degradation after this commit and the memory consumption would be more than accepted in the future. Thus, they have decided to failed the test case even though KPIs were in the acceptable boundaries at the time. These scenarios are two examples of how the test analysis process is dynamic and requirements are prone to change.

The results show that missing and outdated requirements pose a great threat for the test analysis automation process. In some cases, since requirements were not specified clearly or outdated, testers and therefore the tools used in automation process did not possess up to date information about the accepted levels of KPIs for SUT. Testers then had to either decide on some values based on their experience or had to contact other stakeholders to get more information. The dynamic and complex nature of the performance regression testing process forces testers to adapt to new situations and update the requirements and expected values for test data themselves. This finding correlates with the results stated by Bjarnason et al. [5], who mentioned that frequent changes in requirements force testers to find and agree upon the correct version of requirements, thus result in additional work and loss of time. As it was observed at the case company, this problem is even more problematic for the initial phases of the automation process, as the basic tool requires strict boundary values to work with. Furthermore, it has to be pointed out that according to Graham [21], even a small change from the requirements perspective can cause major workload for the testing process. Any changes in requirements essentially impacts the testing phase, as acceptable values for SUT would then be affected.

"Crying wolf" effect. The tool at the case company can sometimes give false positive or false negative verdicts about a certain test case. Test architects and testers have agreed that false positive cases happen more often that false negatives. Three participants have stated that when these mistakes start to happen regularly, people responsible for investigating the failed test cases can lose motivation and concentration. A software designer has stated about this phenomenon:

> *"If you always get "Fails", you lose confidence in verdict tool, you lose motivation to carefully take a look into the problem."* — Software Designer

The result of this study indicates that accuracy of the verdict system is a major challenge. In general, all of the interviewees who have sufficient technical background have stated that the level of accuracy had room for improvement. However, interview subjects had different levels of satisfaction with the accuracy. Some interviewees have stated that they do not trust the results of the verdict system and added that it lacks statistical intelligent. Other subjects have claimed that while they can not trust it completely, it is still at an acceptable level. It can be said that accuracy is an important part of the analysis process. Several researchers have validated their results in similar studies according to their accuracy, as well as investigating different ways to improve it [18] [2] [28]. There are two kinds of errors when a verdict is inaccurate; false positive and false negative. A false positive error in verdict system indicates that the verdict system have analysed a test run and claimed it that there is something wrong and fails the test case when in fact it was acceptable. On the other hand, a false negative error in verdict system happens when an error is not detected but the test run is passed. Subjects have stated that first kind of error was much more frequent than the other kind, but not as damaging. Since it is possible to forward the mistakes and errors towards the customer, testers would like to avoid second kind of errors at all cost. The negative aspect of the first type of error is the “crying wolf” effect. When the verdict system indicates a large number of false positive errors, testers begin to overlook some of the errors thinking that they might also not contain any errors, which might result in passing tests that have serious defects. Similarly, accuracy of locating errors in SUT has been pinpointed as a problematic area by Foo et al. [18] and a solution was proposed to overcome this obstacle.

Several interpretations. There is a common pattern that is found from the interviews held with people from other departments and companies. It can be seen that there is a lack of consensus about the naming of the corresponding system at these companies and departments. The term “verdict” is only used at a single department the case study was held, as other departments and companies had different concepts and solutions for the analysis process. A technical expert that works within the automotive industry has stated that within their system there is not a corresponding tool:

> *“There is no specialized verdict system now, you find it as a test analysis, test metrics and test statistics.”* — Technical Expert

Others have regarded the problem as a part of the test execution process and did not treat it as a distinct and separate part of the whole testing analysis system.

Findings of this study point out that there is a lack of consensus and confusion on the naming, responsibilities and processes between different companies and departments about performance regression test analysis. Each different company that was part of the study had a different views on the problem, even though except for the case company, manual test analysis was used for performance regression analysis. While for the case company verdict decisions on test cases were done towards the end of whole testing processes with the help of the verdict system in test analysis part, another company was using manual labour and considered the process as a part of execution phase of testing. Furthermore, some of the companies that were interviewed and one other department at the case company, considered the test analysis phase as how SUT might perform in the future based on the historical trends. This highlights a contrast with the approach at the department which this study was held, as validation of the test cases are considered as the test analysis process. Lack of consensus about the test analysis process, and companies' reliance on manual test analysis processes inhibit the progression of performance regression test analysis automation.

5.1.3 Tool

There is a tool called verdict system that is developed within the case company to help automating performance regression test analysis. The challenges regarding this tool can be found below.

Usability. One of the issues regarding understandability was that checkpoint creation within the system was rather confusing for some of the interview subjects. Two of the participants have stated that they have encountered problems when testers wanted to create checkpoints. One participant has mentioned that one of the problems with creating checkpoints is that the responsibility of maintaining is unclear after the creation. Same participant also added it should be "easier to define checkpoints", if the department wants the tool with better usability.

One participant has indicated that they had to make extra clicks to get some information about the system, while it was possible to gather this information with fewer actions. It has been stated that:

> *"You get the summary, but since on the top-level you only get this very cryptic checkpoint name, so it is just letters and numbers. You have to expand every checkpoint that you are interesting to see the description that tells you what is checkpoint is doing and trying to find some more information."* — Product Owner

Eliminating the need for expanding to be able to see the information can result in fewer actions to be taken and result in better efficiency for the product.

Usability issues that participants complained about include operability of the tool used for test validation. Efficiency of use was a weak point of the tool according to one interviewee who has explained that it took more steps than necessary to complete a task. This additional burden might not indicate a huge problem in short term, but it distracts the user and causes loss of concentration. Indication of errors were a problematic area as well according to users. Verdict system does not give clear information about the faults, instead it gives codes that are abbreviations. This approach causes for loss time as testers have to figure out what the abbreviations mean. It is possible to experience errors with even the most carefully designed system, but failing to give adequate information about them restraints the recovery process. Other issue about the usability aspect is checkpoint creation for the tool. Interview subjects have stated that it was difficult to create checkpoints and there were uncertainties about how they should be used. These two points also highlight that there might be a lack of clarity about the expectations from the system.

Limited capabilities. Two of the interview subjects mentioned that the tool was unable to give fast feedback. It was complained about how the results or warnings were not presented to users until the very end of the process is completed, which can take 1-2-12-24 hours depending on the process. An operational product owner has stated that:

> *"Well, it is also a really matter of feedback time also. On an hourly basis we run a test and within one hour we get a verdict that says: Hey, something is wrong."* — Product Owner

It has been stated that in order to get a feedback from the system, the system has to complete the whole process before notifying the user of the findings. The users were complaining that they had to wait until the very end of the process in order to get the verdict. This approach causes loss of time which could be easily avoided according to interview subjects.

When a problem happens within the system, it is presented by the system with less than enough information according to one participant. This results in additional workload for employees who are responsible for solving the errors. A function tester has explained by the lack of information about the errors cause them extra work and no indication about it prolongs the solution phase by saying:

> *"They get a fail and send it to us telling "Please take a look". Then, we sit there, saying "OK, what's the problem?"."* — Function Tester

Another soft spot for the current tool used at the case company is detecting constant degradations in the test results. The verdict system does not have the functionality to detect a trend of degradation in test results. It was stated by the participants that a human observer can see the that with each new test result a certain KPI is performing worse than before. Like other problems faced, a human intervention can help detecting this phenomenon before it becomes a major issue for the product and start an investigation to find the root cause of why each test artifact is performing worse than previous one. The time lost to find the root cause for this problem has a negative impact on the department as it becomes more challenging the spot the exact piece of code that made the system perform this way and more people have to devour their work and time to find the specific built.

One problematic incident that the current tool fails to comprehend is the sharp changes with different builds. A test case is not examined thoroughly if the tool passes that case, it is usually only looked into if it fails. Examining a failed test case can be problematic; while it is the last implementation that made the system perform worse than the expected level of quality, most of the time the hidden and main reason that a test case fails happens long before the last commit. While a human can see the exact point of sharp degradation while checking the previous performance of the product and understands which commit had the worse impact to the system, the existing tool fails to notice this occurrence.

A similar case exists for the opposite direction as well. Improvements in performance are usually met with satisfaction, but an unexpected sharp improvement in performance can be the result of a malfunction in the implemented code. Again, the tool passes this test case, while a human can understand the system is not performing as it should do and decides to further investigate it.

According to the results of the study, the tool had limited capabilities and could not perform some of the expected tasks. The interviewees described that it was not possible to understand the root cause of the problem through the system. The tool only gives the information if there is something wrong according to test results but then it is testers' responsibility to investigate the logs in order to find which specific commit caused it. This task becomes even more challenging as the time between two performance regression test runs increases. Testers have to locate a certain commit that caused SUT to perform below the expectations among thousands of other commits. This problem has been also stated by Nguyen et al. [42], who stated that per-checkin performance tests are generally not feasible in large scale industrial settings and determining the cause of a regression is often a time consuming task that depends on the experience of the testers and the complexity of the system. It is also not possible for the testers to get the error messages until the system completes its run. This results in loss time for testers as they could begin investigating if system prompts the error message once it is found.

Another finding of the study was how tool was unable to detect potential risks within the test runs and its inability to warn the testers. Discussions with test architects and developers have pointed out that it is possible to see the potential regressions in performance when the KPI graphs are analysed manually. This situation results in performance degradations, which could have been easily avoided, and forces developers to commit a large proportion of their time to figure out what is wrong with the system. As it was pointed out by Boehm et al. [6], the cost of fixing errors grows larger when they are later discovered. Heger et al. [23] have also stated that software engineers need to manually investigate the root cause of the regressions themselves.

Lack of intelligence. It has been stated by four interviewees that the tool at the case company was lacking any kind of intelligence and was just a simple tool that checks if test results exceed the given limitations. A function tester has stated that:

> *"Verdict should take some of the work but verdict system is too stupid. You put these triggers which indicates capacity decreases and the process behind them are bad and very static."* — Function Tester

It is implied that the system just does what it was designed to do, but nothing more. The tool does not analyse or give any kind of warning to the testers, it just indicates if the test results are within previously stated boundaries or not.

The participants have stated that the verdict system is just comparing the values, the test data value and the boundary value, while presenting the tester last five result of verdicts and test data values. Intelligence, especially the statistical intelligence is missing from the tool, which is desperately needed according to three of the interview subjects. The system at the case company can not distinguish actual problems within the SUT between core dump and test environment problems due to its simplicity. This lack of sophistication in the tool possess one of the biggest treats for the automation process, as responsibility to make these decisions essentially lays on the testers shoulders. It is impossible to further automate the analysis process without tools using statistics, calculating and learning from history in the beginning and later making better decisions.

Documentation. During the documents study it was understood that there was not enough information or a guideline regarding code of the tool used at the case company. It was known that the code was written in Python, and there was a short documentation about the code itself, but since the code was written in an advanced manner, it makes it difficult for the new employees that can be unfamiliar with the level of code to develop new features or make changes on the code. A system test specialist has said that:

> *"It is kind of hard to maintain I think, not hard to maintain, but it is advanced and advanced code and a bit hard to read. So, it is a tough job and if you are not extremely experienced in Python it is challenging to understand some parts of it."* — System Test Specialist

The subject went on to claim that the lack of comprehensive documentation about the code itself was inhibiting the new developers to make changes when they start working in the department.

One participant has explained that there was no information about the certain aspects of the tool. The participant criticized the system on that the users have no idea about how to interpret outputs created by the system by saying:

> *"The user documentation I guess does not even exist. It basically says around the analyze command. It does not say how to what the output means."* — System Test Specialist

System test specialist also added the following as the explanation of the occurance:

> *"If I remember it right, we have not really prioritized documentation and I guess the reason is that nobody has asked for it basically."* — System Test Specialist

Lack of guidelines made it difficult for every tester, but it was even more difficult for new employees to understand how to use the system. Furthermore, along with the complexity of the source code, lack of comprehensive code documentation inhibits necessary improvements to the system. According to the system developers, new developers find it demanding to understand and change the code of the system. The complexity of source code corresponds with the usability of the code documentation. During the writing of the code documentation, it might be overlooked that developers who have inferior ability and less experience need clear instructions and guidelines for future improvements. Similarly, De Boer and Van Vliet [10] states that usability of a documentation is determined by the proximity of the authors' intentions to the expectations of the potential readers. The gap between these intentions and expectations can result in lack of understanding for the source code.

5.1.4 Test Execution Phase

The results of this study indicate that the analysis phase of testing process has affected by test data available. The challenges that have arisen due to test execution phase.

Unstable test data. One of the biggest challenges for performance regression test analysis is caused by unstable test data. The accuracy of measurements is crucial to success of the analysis phase, as false performance indicator data can have a big effect on tools' accuracy as well. It has been reported that for some cases during the test execution phase, it was possible to get different results even if all the conditions were kept unchanged. A systems developer has stated that it was possible to get inconsistent test results:

> *"When running a test case it passes the first time and fails the second time even if we don't change anything. We can get different results and we don't know why. I'm not sure if a tool can understand this problem. It depends on the chance most of the time. I think we have to compensate in the end with the tool."* — Systems Developer

This situation implies that there is an underlying problem with the test environment and it also affects the analysis phase. The tool used at the case company can not differentiate the faulty measurements from the accurate ones, although a manual analyst can easily detect the false one.

Having faulty test data can impact the accuracy of the analysis tools. Since the tool used at the case company only takes last five KPI values during analysis, this approach is seriously affected if a test of certain built is faulty or have crashed during the testing phase. These are called outliers and having an outlier as one of the last five reference points alters the ability of the tool to give more accurate verdicts.

Apart from the outliers, fluctuations also cause serious problems for accurate analysis. Fluctuations in test results for a certain built can happen due to several reasons, one of them being the complexity of systems under test. Test environment and testing tools are mostly accurate within certain limits, but these hint of inaccuracies can cause the automated tools to fail a test case that should meets the requirements in previous and later executions, but performs subpar on a couple test runs. These problems can be detected by a human, but the tool is unable to do so.

Different configurations. In some cases, SUT is tested in different environments with different configurations. As a result, it is possible to get several different test data for the same product that is being tested. This situation creates a challenge for the analysis phase as a decision has to be made accordingly. It is possible to detect these anomalies with manual analysis, but the automated tool used at the case company can not differentiate different configurations and use that information to make a decision about the test case. A function tester has stated that this situation creates additional workload on the testers:

> *"The testing tool does not know if it runs on hardware configuration A or B, and you have to make a lot of investigation."* — Function Tester

Different test configurations affect test data and thus affect the automation process as data available becomes less predictable. The tool can not distinguish the performance of SUT itself and performance improvements or declines caused by test environment according to the interview subjects. Similarly, Foo et al. [18] point out that, incorrect conclusions will be being about the quality of a system, if analysis techniques that cannot differentiate between actual performance regressions and performance differences caused by different environments. In case company's case, this added up to the low accuracy of the tool and puts more unnecessary work on manual analysis.

5.2 RQ2 - Suggested automated performance regression test analysis improvements

This section answers the research question related to automated performance regression test analysis improvements. This study has found 9 practices that can potentially improve the automated performance regression test analysis. The mapping of these practices to the challenges they are found to address are shown in Table 5.1. These practices are described in the section below.

Challenge	**Improvement suggestion**
Unclear expectations	Clear Goals with Rationales
Insufficient requirements	Clear Goals with Rationales
”Crying wolf” effect	Intelligence Component, Historical Data
Several interpretations	Clear Goals with Rationales
Continuous integration	Standardization and Centralization, Testing Map
Lack of standards and central usage	Standardization and Centralization, Testing Map
Branching	Testing Map
Difficulty of root cause analysis	Testing Map
Usability	New UI for Creating Checkpoints
Limited capabilities	Snapshots and Early Alarm
Lack of intelligence	Intelligence Component
Documentation	Improving Documentation
Unstable test data	Intelligence Component, Historical Data
Different configurations	Historical Data

Table 5.1: Mapping of improvement suggestions to the alignment challenges

5.2.1 Infrastructure

Infrastructure improvement. Three interview subjects have argued that standardization and centralization could improve usability for the verdict system. Having a consistent standard with the logs, test cases and documentation would increase usability compliance and common understanding. A technical expert has suggested that:

> *"If you standardize the way of describing certain data, if you want the easier performance, standardize how to write a log file, so you can search for patterns that could be easier. Because you want to identify, let's say you have 50 different computer systems and you want to trace the CPU performance for all those and you have no clue how to find that, then you could give them the same pattern, the you can track them."* — Technical Expert

It can be also argued that by assuring standardization, testers can become more capable to locate issues and troubleshoot.

Centralization also plays an important part in standardization process. Many interview subjects complained about not being able to use some of the tools, test cases and checkpoints because they belong to only a single person or a group. A system test specialist has suggested:

> *"I think it is better to have checkpoints and they belong to everybody. So, we can just for pick any of them, except that we are not using it like that actually."* — System Test Specialist

It can benefit the whole group if there was a centralized repository where all test cases, checkpoints and different tools were collected and belong to everybody. This might improve the capabilities of some of the testers who previously did not have access to these possibilities.

Testing Map. Four interview subjects have stated that having a testing map could make it easier to find the root cause of the errors. A chief leader for continuous integration has stated that:

> *"We call this testing map and we also have tests that we need to combine with resources from different test paths and link these to different releases, because a feature can be released in many different products and tested in test paths and these two need to be linked together. So that we can see a complete picture of a feature on a given release. That's one thing we need to work on also."* — Chief Leader for Continuous Integration

The lack of information about the connection between a certain test case and the branch which the SUT was built upon inhibits the testers' ability to find out how and why the test case had failed. Storing this piece of information and presenting it to the tester for each test case would improve the usability of the verdict system and makes it redundant to use other tools for the testers. It has also been stated that the information about the test cases corresponding to a certain core dump does not exist. A verification engineer has suggested that presenting this piece of information could benefit the system.

5.2.2 Understanding

Clear Goals with Rationales. It has been stated by four participants that the goals and expectations for the processes and tools have to be discussed and agreed upon by all of the stakeholders. Unclear goals result in confusion for developers and testers, as expectations and responsibilities become unclear. Tools used are also criticized for not being dynamic enough and lacking functionality that makes it easily adaptable to changes. Interview subjects have mentioned that requirements were not dynamically updated and clearly defined in some cases and one way to solve this could be introducing rationales. Linking rationales with both the goals for the tools and requirements can help the team to be more blending to unforeseen challenges. A software developer has stated that accuracy of the tools used depend on clear expectations and goals:

> *"My opinion of this verdict tools is that we need to set very clear expectations, so that this system can judge if a checkpoint is passed or fail. This is the reason why it is not very dynamic toward any change or anything else in the product can make it fail."* — Software Developer

It has been mentioned that both expectations of the tools and performance of these tools can be improved with up-to-date requirements, rationales for decisions, clear goals and expectations.

Two participants have also stated that understanding the needs and expectations are essential for designing an automated solution for performance regression test analysis. It is critical to understand what the customer and stakeholders expect and use that information to validate the test cases. A quality assurance engineer has explained that understanding why we need the data in first place is also crucial:

> *"You need to understand what kind of data we are working on, in what context it should work, what is the expectations of the users."* — Quality Assurance Engineer

In order to reduce the misunderstandings and set corresponding goals and expectations for the tools, it is essential to have an open line of communication between testers, developers and other stakeholders.

Complex nature of SUT and test environment at the case company forces testers to make quick and critical decisions and be adaptable to sudden changes. Participants have stated that in order to manage the changing expectations and be adaptable, every tester should be comfortable with the dynamic nature of the test analysis phase. One way to achieve this is to define goals along with the rationales which are agreed upon the beginning with all of the stakeholders. Rationales would help individual testers to make quick decisions on the spot and as the next step would provide more information to implement into the tools for the automation process. This finding drew similarities with the study conducted by Bjarnason et al. [5], who stated that requirement rationales can help the synchronisation by better supporting passing the responsibility between the different roles, in this case to the testers. Requirement rationales would help testers be more proactive in their approach when test cases fail. Currently, testers let developers handle some of the responsibility to find the root cause when a test does not pass. This behaviour puts unnecessary workload on the developers as in most of the cases the test fails due to external causes, not because SUT was not performing up to standards. Agreeing upon initial goals and having requirement rationales might help reduce this problem, as testers would start making decisions on their own. Then, these decisions can be passed on the tools which can help automating the test analysis process.

5.2.3 Tool

Tool improvement. Three interview subjects have mentioned that creating a new checkpoint or editing an existing one has to be made easier. It has been argued that while it is easy to use an existing checkpoint as a template is easy, inserting some new additions such as constraints to that template makes it more difficult. Also, in order to create a new checkpoint a tester have to read upon the corresponding API which also makes it challenging. A software developer has suggested that by building a new interface for checkpoint creation, time spent on this activity could be lowered and new employees can easily use create checkpoints without needing a training or study session. The testers can use this interface to create or edit checkpoints much easier than before.

One of the most complained issues about the verdict system has been about late feedback. The testers have to wait until the testing process is completed to verify a test case and it is not possible to do a real time verification. A quality assurance engineer has suggested that having a snapshot of the process can help save time:

> *"One weakness I feel right now is that we lack the ability to verify the test while it's running. We lack real time verifying test results. If we divide in the middle, we can have take some snapshots and see if it's broken, but right now we don't have that."* — Quality Assurance Engineer

Receiving early feedback can be critical and save some time during the process. A change manager has stated that having quick feedback loops could provide more information in acceptable time frame, thus allowing testers to know if the changes in the system have caused any problems.

Two interview subjects have stated that improving documentation for several different components could improve the general usability of the verdict system. Having a consistent documentation throughout the whole system would increase consistency and understandability. This certainly applies for the code documentation. A software test engineer has mentioned:

> *"It can always be better documented. The code is written in Python and it has a few syntaxes that are on a very high or an expert level. It is not very easy for beginner Python programmers to understand how it works, so that is perhaps something that could be improved."* — Software Test Engineer

Lack of comprehensive documentation on top of code written at an expert level makes it challenging for new employees to grasp how verdict system works. A technical expert have suggested that test cases also need to be written in a more descriptive way. Again, this would allow new testers to grasp how certain test cases work, thus easing their integration into the testing team.

Intelligence Component. One common point of suggestion to improve accuracy is integration of a statistical intelligence component inside the verdict system. A function tester has stated:

> *"In my ideal world, we should be able to produce a verdict system that sorts out this stuff, with the help of statistics. It doesn't have to have the intelligence that we have but it is very easy to see this problem."* — Function Tester

The verdict system fails to see degradations which a human tester have no problem seeing and detect future regressions for the SUT. A software designer has suggested that using historical data along with statistical analysis can help detect performance regressions:

> *"In my ultimate world, you will make a verdict script that can filter out the noise and trigger the degradation that is statistically proven to be a degradation based on history."* — Software Designer

A verdict system with statistical intelligence can have more accurate verdicts and at the same time warn testers about abnormalities which can cause problems in the future. These abnormalities also include expected degradations. It was mentioned that sometimes degradations are wanted and expected, especially if new functionality has been recently delivered for SUT. With statistical analysis verdict system could detect if there is no regression in performance, which could also be troubling.

5.2.4 Test Execution Phase

Historical Data. Most of the interview subjects have suggested that in order to improve accuracy for automated performance regression test analysis at the case company historical data has to be used more extensively and a statistical intelligence component has to be integrated within the verdict system.

The verdict system at the case company only takes the last 5 result into account during the analysis process. A function tester has stated this was not enough and more data should be used when making a decision on a test case:

> *"If there's a history, checking last 5 builds is ridiculous. We rather look at the last 400 builds."* — Function Tester

A systems developer has also added:

> *"Why not use the last 50 instead of 5, and you can filter out the noise and find some trend. You can find two channels, or you can see a slow trend that is going down. However, the last five is not really relevant."* — Systems Developer

Taking more than previous 5 test results into consideration could provide more capabilities and used in a way to warn testers in case of unexpected regressions with the indicators. Using more historical data can help to make sense of the existing test result data and reduce the false verdicts.

6

Discussion

This chapter discusses the the results presented in the previous chapter and relates it to research literature. Section 6.1 discusses the first research question, challenges regarding automation of performance regression test analysis. Section 6.2 discusses the second research question, improvements. Section 6.3 discusses the future work for the research. Section 6.4 discusses validity threats of the study. Implication for practitioners and contribution to the academic research is discussed in sections 6.5 and 6.6.

The aim of this study was to identify the challenges regarding automation of performance regression test analysis and suggest possible improvements for industrial applications. In order to achieve these goals, a case study has been conducted, 12 semi-structured interviews were held and several other data collection methods were used to obtain data.

6.1 RQ1 - Challenges regarding automation of performance regression test analysis

In relation to RQ1 - challenges regarding automation of performance regression test analysis, the results include: 1) Infrastructure has to be set up in a way that encourages and makes it easier to automate the process. Absence of such platform hinders the automation process. Furthermore, continuous integration makes it difficult to assess the test results as they become inadequate quickly. 2) Unclear goals and insufficient requirements slow down the automation process. There is not a reached consensus about the concepts regarding performance regression test analysis within the industry. 3) Tool used at the case company is a simple, useful tool, but it has serious limitations. Lack of intelligence and usability prevents from achieving full potential of the tools. 4) Data produced after test execution can be unreliable which makes automation process difficult. Due to variations of different hardware that are used successfully evaluating a test case is challenging.

6.2 RQ2 - Suggested automated performance regression test analysis improvements

The findings regarding RQ2 - suggested automated performance regression test analysis improvements, the most conspicuous results are: 1) Goals have to be stated, along with their rationales. As the first step towards full automation, testers need to understand the reasoning behind the goals to be more dynamic. 2) A more centralized infrastructure with clear standards can help the developers and testers to make full use of their tools. 3) A testing map could improve the understanding regarding the connections between each build and new developments. This in turn can help a better assessment of the test results. 4) Tools for automation process has to be designed in a way that allows testers to be more dynamic and flexible. An early alarm system that includes warning as a result apart from pass and fail would help in some cases, such as core dumps. 5) An intelligent component is necessary for the next step in the automation process. Tools have to make decisions on their own and it is only possible with an intelligent component. 6) Making use of historical test data is highly critical as in order to make better decisions and have more accurate verdicts tools have to take advantage of a rich source of historical data.

6.3 Future work

Improving the tool with respect to its accuracy and usability within limited amount of time and budget is a major focus for the case company. After studying the results of the study, researchers and industrial supervisors have agreed that improving the intelligent ability of the tool would be the most practical and efficient solution with the available time and resources. Introduction of a statistical component into the verdict system can provide more accurate verdicts to the testers by making a better usage of the vast source of historical data available and eliminate the heterogeneous environments problem. In this section, the method that has been used within this component is described along with how it would can be used within the verdict system as a component.

There have been research done regarding historical data and it is possible to use this major source of information along with association rules as a part of the verdict system. Even though making use of historical data does not guarantee better accuracy directly, it provides valuable information to the testers about how a certain test case was handled in previously similar scenarios. Furthermore, as time passes each verdict and test result can be used to create association rules that offer even more information as a part of verdict system. These data and rules can act as an early warning system for testers for critical or previously unforeseen situations. For example, even though the CPU usage for the SUT is lower than the threshold value for a certain test case, and thus has to be set as passed, the improved verdict system can warn the tester about how the CPU usage should be much lower than it actually is according to all the other available test result data. Testers then can use this warning as an indication of underperformance of the SUT an let the developers aware of the situation. Information about these methods and how they would fit into the current structure are given below.

Deriving knowledge from patterns that exist in the huge amount of data is known as Knowledge Discovery in Databases (KDD), which is a remarkable and extensive research field that has a lot of consideration from the research community [60]. According to Fayyad et al. [16] the data mining assignments utilized in KDD process can be grouped into six distinct categories, which are summarization, clustering, regression, classification, dependency modeling and change and deviation detection. The majority of these categories address the issues related to ones occurring while analyzing performance tests [60].

The predominant complication when mining performance testing datasets is formulated as obtaining the level of correlation between the datasets that contain performance observations. The performance data either belongs to a baseline dataset or to a target dataset. A comparison between the values of the performance counters collected during the target run with the values of the corresponding baseline runs has to be performed, to determine if a target run was successful or not. If the values are similar, the target run should have acceptable performance. If the baseline and target datasets diverge substantially, the target run has different performance which must be documented for further investigation [60].

The dependency modeling category should be chosen on mining performance testing data, because the function ability of this category is to describe significant dependencies between variables [31]. Nowadays, the dependency modeling category is commonly referred to as association rule mining, which is a term that was introduced by Agrawal et al. [1].

By applying association rule mining on prior test runs, a set of association rules can be extracted. An association rule comprises of a premise and a consequent. The rule expresses that if the premise is true in a new test run, then the consequent will also be true with high probability. Figure 6.1 demonstrates an example of an association rule that expresses "if the CPU and memory utilization are examined to be at the low and medium levels respectively, then the throughput will be at the high level" [18].

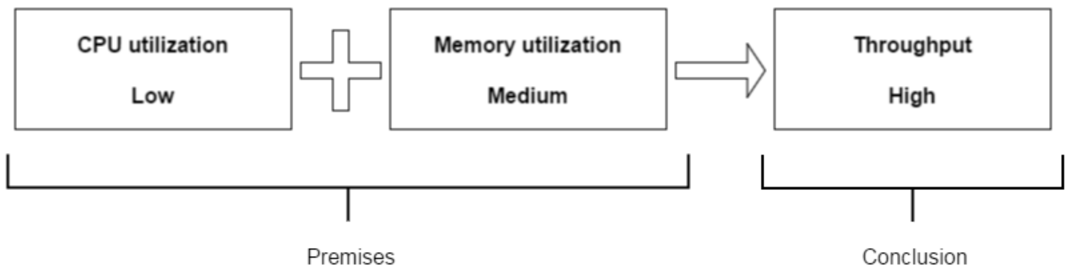


Figure 6.1: Thematic map of challenges

The probability with which an association rule applies is described by its support and confidence measurements. Support evaluates the ratio of times the rule applies, namely the counters in the premise and consequent are examined collectively with the specified values. For example, low support indicates that the association rule might have been discovered coincidentally. Confidence evaluates the probability that the premise of the rule implies the consequent, namely how frequently the consequent is true when the premise is true. For instance, if the confidence value of the rule in Figure 6.1 is near one, then it signifies that when CPU and memory utilization are examined to be at the low and medium levels respectively, throughput will be at the high level most of the times [18].

The program flow of the new verdict system will be similar to the existing one, with small differences. New analyze function would take all of the previous tests results as inputs and form expected results for each separate KPI. It is possible to use Orange, an open source toolbox written in Python, for both data mining and creating association rules [11]. Orange is able to create association rules for an existing data set, which can be imported as a Excel file or tab delimited data file. Testers then can compare the expected KPI values with the actual value of the KPI according to the test results to confirm that there are no irregularities with the results.

Furthermore, this case study is only focused at a single, large company. It would be of value to involve multiple companies with different sizes. This could help understanding if the findings can be observed in different companies, thus generalized, or unique to the case company. Moreover, The data was obtained through semi structured interviews. Conducting a statistical analysis at the case company would help understand precise success of the verdict system and reveal the challenges in a quantifiable form. Finally, it might be beneficial to apply the suggested improvements and stated algorithms to the current problem observed at the case company to understand if it is possible to improve the current situation.

6.4 Validity Threats

The validity of a study indicates the trustworthiness of the results and findings to what extent the results are true and not biased by the researchers' subjective point of view [48]. Validity threats in this thesis study were examined according to the classification made by Runeson and Höst [48] in their guideline and it covers construct validity, external validity, internal validity and reliability threats. These threats, along with the approaches and methods to reduce them, were investigated and explained in this section.

6.4.1 Construct Validity

Construct validity refers to whether measurements actually model independent and dependent variables from which the hypothesized theory is constructed [59]. A study has a high construct validity if the theory is constructed in a way that used parameters are relevant to research questions.

In order to limit the threats against construct validity, both qualitative and quantitative data methods are used to elicit information. Furthermore, interview subjects are informed about the study beforehand, and a general idea about what the research is about has been given, Interviews are drafted with the supervision of 4 experienced people in their fields; 2 from academia and 2 from industry who gave feedback before the interview template was finalised. Interview subjects are reminded of their anonymity and confidentiality before each interview to prevent eliciting false and unreliable information. After the interviews are complete, a copy of both the interview recordings and transcripts shared with the subjects to confirm that there is a consensus on what was meant by the researchers and what is understood by the subjects and to avoid misunderstandings.

6.4.2 Internal Validity

Internal validity focuses on how confident researchers can be that the measured variables actually caused the outcome [17]. In order to establish internal validity, researchers have to show that the outcome was the result of the factors stated in the study.

Triangulation method can be used as a tool to increase the precision of empirical research, as applying it means taking different angles towards the studied object and thus providing a broader picture [48]. Within this study, researchers have chosen the triangulation method to cover more than a single aspect of the study, and ensure that the collected data reflects the conclusions that are drawn at the end. Using different data sources, a combination of both qualitative and quantitative data collection methods such as interviews, surveys, system analysis, workshops and documents study, aimed to limit the disadvantages of each method and therefore, reduce the effects of the validity threats.

6.4.3 External Validity

External validity refers to the applicability of the research to domains other than the one that is under observation (Wright, Kim and Perry, 2010). It is about whether the results can be generalized outside the scope of the study [17].

One of the biggest concerns during the study was to ensure that the findings and results of the conducted research can be applied to different domains other than one at the case department. In order to reduce this threat, people from other companies and departments were included in the study as participants and their feedbacks are highly regarded. Furthermore, while case department specific details are being investigated, problems faced within the same context with the other companies and departments are also studied. All of these different sources of information was taken into account while designing and implementing a solution for the problem.

6.4.4 Reliability

Reliability is concerned with to whether the data and the analysis are dependent on the specific researchers [48]. In order for a study to be reliable, the results should not differ if it is repeated by different researchers.

Since this study was conducted by two researchers, it was possible to avoid single researcher bias. Collected data is checked and analysed by both of the researchers and sent to subjects who had time to review it to ensure they were correctly transcribed.

6.5 Implications for practitioners

This study presents a set of challenges regarding automation of performance regression analysis in a large company and proposes improvement suggestions for these encountered challenges. Prior to this study the case company was not confident in verdict system's performance. The verdict system was developed without an extensive research simply to reduce some of the mundane work for the system testers during the analysis phase. The results of this study helped the case company to make an objective assessment of the verdict system. Four different categories were highlighted as problematic areas and improvement suggestions to overcome these problems were proposed. This study presented a broader understanding of their tools and processes for the case company and helped them evaluate their solution.

The four categories where the challenges identified were infrastructure, understanding, tool and test execution. Along with the suggestions each of these problematic areas can be better understood and improved. By improving the current state of the analysis process according to results of this study it is possible to have more accurate verdict results and fasten the analysis process. Also, it can be possible to make the verdict system easier to use for both experienced and new users of the system. As a result of implementing these suggestions, the analysis process can be automated even further and a faster and more accurate analysis phase can be achieved. Testers can control a greater extend of the test cases, thus ensuring a higher level of quality for the product. Developers can avoid investigating unconfirmed bugs and regressions and only focus on debugging and optimizing the code they are responsible of.

Even though the challenges regarding automation of performance regression analysis were stated, the results section of this thesis study has also confirmed that the verdict system is a highly useful tool for testers and helps eliminating mundane work within the department. Each interviewee from the case department has agreed that having verdict system makes sense for the company and in some cases it would have been impossible to achieve the department goals. Prior this study the test architects were unsure about the future of the verdict system and discussing how the department should proceed. Two options were considered as the future tool that was going to be used in the case department; either verdict system will be rewritten from the beginning and all of the advantages (as well as the disadvantages) of the old tool would be discarded in favour of creating a better system, or verdict system will be kept and it would be improved even further. This study provided necessary information for the test architects to make the decision about the path for the department.

Practitioners outside the case company such as different organizations and companies may use the results from this research to understand the potential problems that can occur during the transition from manual processes to automated non-functional test results validation. By considering the key points stated as the results of this study, organizations and companies can benefit during constructing their own automation of performance regression analysis processes.

6.6 Contribution to academic research

This study adds to the existing body of academic knowledge within software engineering that is focused on software testing. This thesis deals with the performance regression testing area which lacked of studies and extensive research.

The focus of this study is non-functional test validation and performance regression testing, which have received less interest than functional regression testing. The challenges during the transitional phase of switching from manual performance analysis to automated performance analysis in a large organization are highlighted. Furthermore, several improvement suggestions have been given to overcome these problems. The results of these studies can also be used in future research to further extend the study and as a base to build upon.

7

Conclusion

Software testing is a crucial part of the software development process that takes nearly half of the time spent on projects. Many organizations have tried to reduce the time and human power spent by automating the testing phase. However, performance regression testing, compared to functional regression testing, has not been the focal point of academic attention and only few studies have been done in this area. The aim of this study was to help filling this gap by identify the challenges regarding automation of performance regression test analysis and suggest possible improvements for industrial applications. In order to achieve these goals, a case study has been conducted at a telecommunication company at Sweden, 12 semi-structured interviews were held and several other data collection methods were used to obtain data. The data then analysed and turned into results. The main contributions of this thesis work are highlighting the problems and suggesting solutions for the practice as well as providing an insight on problems faced during performance regression test result analysis automation for the research.

The results of this study indicate that these challenges can be listed under four main groups; infrastructure at the company, understanding differences between involved actors, tools used for the processes and data available to make analysis. To overcome these challenges several improvements are suggested. These six suggestions are; stating clear goals, centralizing the infrastructure, creating a testing map, improving the tools, adding an intelligent component and making use of historical data as much as possible. As the next step in the future, out of these six improvement suggestions two of them has been selected as the core and a proposal has been created upon them. This proposal can be used at the case company as future work to improve the current state of automation of performance regression test analysis.

In conclusion, the challenges regarding performance regression test analysis automation have to be highlighted and studied in order to improve the current state of the process.

Bibliography

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[2] Marcos K Aguilera, Jeffrey C Mogul, Janet L Wiener, Patrick Reynolds, and Athicha Muthitacharoen. Performance debugging for distributed systems of black boxes. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 74–89. ACM, 2003.

[3] Earl Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Sang-Im Yoo. The oracle problem in software testing: A survey. 2015.

[4] Antonia Bertolino. Software testing research: Achievements, challenges, dreams. In *2007 Future of Software Engineering*, pages 85–103. IEEE Computer Society, 2007.

[5] Elizabeth Bjarnason, Per Runeson, Markus Borg, Michael Unterkalmsteiner, Emelie Engström, Björn Regnell, Giedre Sabaliauskaite, Annabella Loconsole, Tony Gorschek, and Robert Feldt. Challenges and practices in aligning requirements with verification and validation: a case study of six companies. *Empirical Software Engineering*, 19(6):1809–1855, 2014.

[6] Barry W Boehm et al. *Software engineering economics*, volume 197. Prentice-hall Englewood Cliffs (NJ), 1981.

[7] Frederick P Brooks. *The mythical man-month*, volume 1995. Addison-Wesley Reading, MA, 1975.

[8] Vittorio Cortellessa, Antinisca Di Marco, and Paola Inverardi. *Model-based software performance analysis*. Springer Science & Business Media, 2011.

[9] John W Creswell. *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2013.

[10] Remco C De Boer and Hans Van Vliet. Writing and reading software documentation: How the development process may affect understanding. In *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, pages 40–47. IEEE Computer Society, 2009.

[11] Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, and Blaž Zupan. Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353, 2013. URL `http://jmlr.org/papers/v14/demsar13a.html`.

[12] Giovanni Denaro, Andrea Polini, and Wolfgang Emmerich. Early performance testing of distributed software applications. In *ACM SIGSOFT Software Engineering Notes*, volume 29, pages 94–103. ACM, 2004.

[13] Paul M Duvall, Steve Matyas, and Andrew Glover. *Continuous integration: improving software quality and reducing risk.* Pearson Education, 2007.

[14] Ericsson.com. Traffic and market report, 2015. URL `http://www.ericsson.com/res/docs/2012/traffic_and_market_report_june_2012.pdf`.

[15] Ericsson.com. Evolved packet gateway, 2015. URL `http://www.ericsson.com/se/ourportfolio/products/evolved-packet-gateway`.

[16] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[17] Robert Feldt and Ana Magazinius. Validity threats in empirical software engineering research-an initial survey. In *SEKE*, pages 374–379, 2010.

[18] King Chun Foo, Zhen Ming Jack Jiang, Bram Adams, Ahmed E Hassan, Ying Zou, and Parminder Flora. An industrial case study on the automated detection of performance regressions in heterogeneous environments.

[19] Marie-Claude Gaudel. Testing from formal specifications, a generic approach. In *Proceedings of the 6th Ade-Europe International Conference Leuven on Reliable Software Technologies*, Ada Europe '01, pages 35–48, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-42123-8. URL `http://dl.acm.org/citation.cfm?id=646580.697772`.

[20] Ahmed S. Ghiduk, Moheb R. Girgis, and Eman Abd-Elkawy. A survey of regression testing techniques. *International Journal of Advanced Research in Computer Science*, 3(5), 09 2012. URL `http://search.proquest.com/docview/1443737357?accountid=10041`.

[21] Dorothy Graham. Requirements and testing: Seven missing-link myths. *Software, IEEE*, 19(5):15–17, 2002.

[22] Heather Harreld. Nasa delays satellite launch after finding bugs in software program. *Federal Computer Week*, 1998.

[23] Christoph Heger, Jens Happe, and Roozbeh Farahbod. Automated root cause isolation of performance regressions during software development. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, pages 27–38. ACM, 2013.

[24] William C Hetzel and Bill Hetzel. *The complete guide to software testing.* John Wiley & Sons, Inc., 1991.

[25] Robert M Hierons. Verdict functions in testing with a fault domain or test hypotheses. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 18 (4):14, 2009.

[26] Alexey Ieshin, Marina Gerenko, and Vadim Dmitriev. Test automation: Flexible way. In *Software Engineering Conference in Russia (CEE-SECR), 2009 5th Central and Eastern European*, pages 249–252. IEEE, 2009.

[27] ISO. Information technology – open systems interconnection – conformance testing methodology and framework – part 1: General concepts. ISO/IEC 9646-1, International Organization for Standardization, Geneva, Switzerland, 1994.

[28] Zhen Ming Jiang, Ahmed E Hassan, Gilbert Hamann, and Parminder Flora. Automated performance analysis of load tests. In *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, pages 125–134. IEEE, 2009.

[29] Todd D Jick. Mixing qualitative and quantitative methods: Triangulation in action. *Administrative science quarterly*, pages 602–611, 1979.

[30] Passant Kandil, Sherin Moussa, and Nagwa Badr. Regression testing approach for large-scale systems. In *Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium on*, pages 132–133. IEEE, 2014.

[31] V Karthikeyani and I Parvin Begum. Comparison a performance of data mining algorithms (cpdma) in prediction of diabetes disease. *International Journal*, 2013.

[32] Jacob Larsson and Markus Borg. Revisiting the challenges in aligning re and v&v: Experiences from the public sector. In *Requirements Engineering and Testing (RET), 2014 IEEE 1st International Workshop on*, pages 4–11. IEEE, 2014.

[33] Timothy C Lethbridge, Susan Elliott Sim, and Janice Singer. Studying software engineers: Data collection techniques for software field studies. *Empirical software engineering*, 10(3):311–341, 2005.

[34] Lu Luo. Software testing techniques. *Institute for software research international Carnegie mellon university Pittsburgh, PA*, 15232(1-19):19, 2001.

[35] Philipp Mayring. *Qualitative inhaltsanalyse.* Springer, 2010.

[36] James D. McCaffrey. Software testing vs. analysis, 2006. URL `https://jamesmccaffrey.wordpress.com/2006/04/16/software-testing-vs-analysis/`.

[37] Jani Metsä, Mika Katara, and Tommi Mikkonen. Testing non-functional requirements with aspects: An industrial case study. In *Quality Software, 2007. QSIC'07. Seventh International Conference on*, pages 5–14. IEEE, 2007.

[38] Matthew B Miles and A Michael Huberman. *Qualitative data analysis: An expanded sourcebook.* Sage, 1994.

[39] Rob Miller. Heuristic evaluation. URL `http://groups.csail.mit.edu/graphics/classes/6.893/F03/lectures/L5.pdf`.

[40] D. L. Morgan. Qualitative content analysis: A guide to paths not taken. *Qualitative Health Research*, 3(1):112–121, 1993. doi: 10.1177/104973239300300107.

[41] Janice M Morse. Approaches to qualitative-quantitative methodological triangulation. *Nursing research*, 40(2):120–123, 1991.

[42] Thanh HD Nguyen, Meiyappan Nagappan, Ahmed E Hassan, Mohamed Nasser, and Parminder Flora. An industrial case study of automatically identifying performance regression-causes. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 232–241. ACM, 2014.

[43] Jiantao Pan. Software testing, 1999. URL `http://users.ece.cmu.edu/~koopman/des_s99/sw_testing/`.

[44] Macario Polo, Pedro Reales, Mario Piattini, and Christof Ebert. Test automation. *IEEE software*, (1):84–89, 2013.

[45] Colin Robson. Real world research. 2nd. *Edition. Blackwell Publishing. Malden*, 2002.

[46] Nicholas C Romano Jr, Christina Donovan, Hsinchun Chen, and Jay F Nunamaker Jr. A methodology for analyzing web-based qualitative data. *Journal of Management Information Systems*, 19(4):213–246, 2003.

[47] Routledge. Analysing data ii: Qualitative data analysis, 2015. URL `http://cw.routledge.com/textbooks/9780415493932/downloads/ch13.ppt`.

[48] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131–164, 2009.

[49] Roger Sapsford and Victor Jupp. *Data collection and analysis.* Sage, 2006.

[50] Carolyn Seaman. Using qualitative methods in empirical studies of software engineering. *Available [online] also at: http://ccsl. ime. usp. br/files/QualMethods% 20minicourse% 20USP-1. pdf [accessed in Malaysia: 7 July 2013]*, 2013.

[51] David Silverman. *Interpreting qualitative data: Methods for analyzing talk, text and interaction.* Sage, 2006.

[52] Sarbjeet Singh, Gurpreet Singh, and Sukhvinder Singh. Software testing. *International Journal of Advanced Research in Computer Science*, 1(3), 09 2010. URL `http://search.proquest.com/docview/1443701695?accountid=10041`. Copyright - Copyright International Journal of Advanced Research in Computer Science Sep 2010; Last updated - 2015-05-21.

[53] Connie U Smith. *Performance engineering of software systems.* Addison-Wesley Longman Publishing Co., Inc., 1990.

[54] Susan Spiggle. Analysis and interpretation of qualitative data in consumer research. *Journal of consumer research*, pages 491–503, 1994.

[55] Matt Staats, Michael W Whalen, and Mats PE Heimdahl. Programs, tests, and oracles: the foundations of testing revisited. In *Software Engineering (ICSE), 2011 33rd International Conference on*, pages 391–400. IEEE, 2011.

[56] Ben Tristem. Usability heuristics applied to modern aviation. URL `http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol2/bvct/`.

[57] Elaine J Weyuker and Filippos I Vokolos. Experience with performance testing of software systems: issues, an approach, and case study. *IEEE transactions on software engineering*, (12):1147–1156, 2000.

[58] James Whittaker et al. What is software testing? and why is it so hard? *Software, IEEE*, 17(1):70–79, 2000.

[59] Hyrum K Wright, Miryung Kim, and Dewayne E Perry. Validity concerns in software engineering research. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, pages 411–414. ACM, 2010.

[60] Z Zaleznicenka. *Automated detection of performance regressions in web applications using association rule mining.* PhD thesis, TU Delft, Delft University of Technology, 2013.