Int J Syst Assur Eng Manag (Apr-June 2011) 2(2):114–125
DOI 10.1007/s13198-011-0065-6

ORIGINAL ARTICLE

Trade-off between automated and manual software testing

Ossi Taipale · Jussi Kasurinen · Katja Karhu · Kari Smolander

Received: 14 October 2010 / Revised: 22 August 2011 / Published online: 7 September 2011

Abstract The study explores the current state of test automation in software testing organizations by focusing on the views and observations of managers, testers and developers in each organization. The case study was conducted in selected organizational units that develop and test technical software for industrial automation or telecommunication domains. The data was collected with 41 theme-based interviews in each unit. The interview data was analyzed qualitatively by using the grounded theory research method. It was found that although test automation was viewed as beneficial, it was not utilized widely in the companies. The main benefits of test automation were quality improvement, the possibility to execute more tests in less time and fluent reuse of testware. The major disadvantages were the costs associated with developing test automation especially in dynamic customized environments. Such issues as properties of tested products, attitudes of employees, resource limitations, and customers influenced the level of test automation in the case organizations.

O. Taipale (✉) · J. Kasurinen · K. Smolander
Laboratory of Software Engineering, Lappeenranta University of Technology, 20, 53851 Lappeenranta, Finland
e-mail: ossi.taipale@lut.fi

J. Kasurinen
e-mail: jussi.kasurinen@lut.fi

K. Smolander
e-mail: kari.smolander@lut.fi

K. Karhu
Bitten Oy, Hämeentie 32 B 68, 00530 Helsinki, Finland
e-mail: katja.karhu@bitten.fi

Keywords Software testing · Test automation · Manual testing · Benefits and drawbacks of test automation · Qualitative research · Empirical study

1 Introduction

Testing work can be roughly divided into automated and manual testing. Automated software testing means the automation of software testing activities (Dustin et al. 1999). These activities include the development and execution of test scripts, the verification of testing requirements, and the use of automated test tools. Testing a software product forms a considerable expense, but so do the costs caused by faults in the software product. For example, in the United States only, the annual costs caused by erroneous software were 59.5 billion dollars in 2000 (Tassey 2002). Torkar and Mankefors (2003) found that 60% of the developers in their survey claimed that verification and validation were the first to be neglected in cases of resource shortages during a project. It is a common view that if some of the test process phases could be automated and the existing processes themselves streamlined, the available resources could be directed towards additional testing or gaining savings (Ramler and Wolfmaier 2006). Berner et al. (2005) estimate that most of the test cases in one project are run at least five times, and one-fourth over 20 times. Especially smoke tests, component tests, and integration tests are repeated constantly, so there seems to be an incentive for automation development.

The establishment of test automation is a risky investment project, however. Persson and Yilmaztürk (2004) note that the decision on what to automate and what to test manually should be defined early in the project to avoid

problems. To support decision making, Ramler and Wolfmaier (2006) have developed a model for estimating test automation costs.

According to Bertolino (2007), test automation is a significant area of interest in current testing research, with the aim to improve the degree of automation, either by developing advanced techniques for generating test inputs, or by finding support procedures to automate the testing process. According to Briand (2007), empirical studies are important for software testing research in order to compare and improve software testing techniques and practices.

The objective of this study is to analyze the current state and use of test automation in software testing organizations. Our research questions are: (1) what facilitates the use of test automation, i.e. what issues explain why test automation is extensively utilized in an organization, and (2) what hinders the use of test automation, i.e. what issues explain why test automation is not utilized at all or minimally in an organization?

We selected five case organizations for the study from 30 organizations that we surveyed in the research project (Taipale et al. 2006a). Organizations for the survey were originally selected from a population of 62 information and communication technology companies. The organizations consist of software development and testing companies belonging to the industrial automation or telecommunication domain. The data was collected by interviewing people from different organizational positions (managers, testers, and developers) in each of the organizations. The data collection included four theme-based interview rounds. We visited the companies and carried out 41 tape-recorded interviews. The themes of each interview round are available at http://www.it.lut.fi/project/anti/.

In data analysis, the grounded theory research method (Glaser and Strauss 1967; Strauss and Corbin 1990) was applied. We selected grounded theory as the research method because according to our knowledge, the practice of software test automation has not been covered widely enough in previous research, and grounded theory has the ability to uncover issues from the practice under observation that may not have been identified in earlier literature.

This study is a continuation of a series of studies with a similar research design. These studies approach software testing practice empirically from various viewpoints, including process improvement (Taipale et al. 2005; Taipale et al. 2006a), schedules (Taipale et al. 2006b), knowledge management (Taipale et al. 2007), and outsourcing (Karhu et al. 2007). An earlier and shorter version of the practice of test automation has been published (Karhu et al. 2009). All of the previous case studies were conducted in the same five organizational units that represent polar points of our original population of 62 organizations.

The paper is structured as follows: first, we introduce related research. The research process is described in Section 3, the analysis results are presented in Section 4, and finally, Sections 5 and 6 contain discussion and conclusions.

2 Related research

The trade-off between automated and manual testing seems to depend on many things. Results of the related studies offer factors that either facilitate or hinder the use of test automation. Usually the first instinct of adopting test automation is just to apply it to do whatever the testers were previously doing manually (Berner et al. 2005). However, automation cannot replace manual testing completely (Bach 1997) or eliminate personnel costs, as there has to be someone who oversees and maintains the test environment and takes care of the test development. Persson and Yilmaztürk (2004) note that the establishment of automated testing may fail if the test automation is not planned thoroughly or if is implemented without an appropriate implementation strategy. Test automation sets several requisites on the project, and in some cases, (Blackburn et al. 2004; Santos-Neto et al. 2007) the outcome of applying test automation has not been useful for the project.

Prior studies regarding test automation in practice are presented, for example, in (Persson and Yilmaztürk 2004; Hartman et al. 2007). In addition, for example Ng et al. (2004) have studied the processes in the Australian software industry. Ng et al. (2004) applied the survey method to establish knowledge on such topics as testing methodologies, tools, metrics, standards, training, and education. The study indicated that the most common barriers to develop testing were the lack of expertise in adopting new testing methods and the costs associated with the testing tools.

Both manual and automated testing have individual advantages and disadvantages. For example, Ramler and Wolfmaier (2006) summarize the differences between manual and automated testing by suggesting that automation should be used to prevent further errors, while manual testing is better suited for finding new and unexpected errors. According to Dustin et al. (1999), reasons for using automated software testing are, for example, that manual testing is time consuming, and that test automation increases efficiency, especially in regression testing, where test cases are executed iteratively after making changes to the software. Kaner (2000) has observed that the automation tools in the graphical user interface (GUI) testing context are mainly used to assist the tester, rather than to automate GUI testing widely. Overall, there seem to be

several reported misconceptions about test automation (Persson and Yilmaztürk 2004; Kaner 2000; Kaner 1997; Berner et al. 2005), leading to difficulties in implementation when automation is introduced to ill-suited areas of application.

The practical applicability of software test automation in specific domains is studied for example in (Persson and Yilmaztürk 2004; Hartman et al. 2007), and industry-wide aspects are covered for example in (Bach 1997; Fewster 2001). In general, the literature seems to agree that test automation is a plausible tool for enhancing quality, and consequently, for reducing software development costs in the long run. Automatic test generation may improve for example test coverage and it is possible to use different techniques such as genetic algorithms for the generation (Mantere 2003). In addition, Ramler and Wolfmayer (2006) discuss the aspect of error prevention in test automation. Kaner (1997) suggests that test automation should be understood more as a quality assurance tool than a testing application, as most of the errors are found when creating the automation tools, not by using them. Therefore, there are several aspects on software test automation that could still be enhanced and studied further.

3 Research process

To understand the current state and use of software test automation in different kinds of software testing organizations, and to explain why some organizations utilize test automation more than others, an exploratory and qualitative strategy following the grounded theory approach (Glaser and Strauss 1967; Strauss and Corbin 1990) was selected. According to Seaman (1999), the grounded approach enables the identification of new theories and concepts, making it a valid choice for software engineering research. Grounded theory was selected as the research method because it offers tools to identify real-world concepts and to build theoretical constructs that can explain them. In this study, we observe the practice and identify how and to what extent test automation is currently in use in the case organizations. We continue this with theoretical explanations of why the organizations are currently utilizing (or not utilizing) test automation. These explanations are based on identified patterns, commonalities, and differences between the organizations. We analyze the interviewees' perceptions of the current state of test automation in their organizations. The choice of grounded theory as the research method is justified because the need for qualitative approaches in the areas related to human behavior is recognized widely also in software engineering research (Seaman 1999; Shaw 2003; Sjöberg and Dybå 2007).

We follow the process of building a theory from case study research described by Eisenhardt (1989) and its implementation example (Pare and Elam 1997). The principles for an interpretive field study have been derived from (Klein and Myers 1999). Other example studies using the grounded theory method are (Carter and Dresner 2001; Smolander et al. 2008).

Our data analysis includes within-case analysis and cross-case analysis (Eisenhardt 1989). Within-case analysis involves case study write-ups for each site. Cross-case analysis includes a search for general patterns. We have selected categories and dimensions and then looked for within-group similarities coupled with inter-group differences (Eisenhardt 1989) in the case organizational units. For example, if an organization identified reuse as an affecting facilitator to their testing process, the corresponding organizations were compared for similarities in this respect to identify a common enabler, while other companies were analyzed to understand why this phenomenon did not occur in their organizations.

3.1 Selecting case organizational units

The standard ISO/IEC 15504-1 (2002) specifies an organizational unit (OU) as a part of an organization that is the subject of an assessment. An OU deploys one or more processes having a coherent process context, and operates within a coherent set of business goals. An OU is typically a part of a larger organization, although in a small organization, the OU may cover the whole organization. The OUs and the interviewees of this study are presented in Table 1. The reason to use an OU as the unit for observation was that we wanted to normalize the effect of the company size to get comparable data. The criticality of the produced or tested software was measured during the first interview round by asking how severe problems faults in their products can cause (Taipale et al. 2006a). The OUs estimated the criticality of the software using a five-point scale from irritation and dissatisfaction to loss of human lives. The objective was to select OUs with an above-average software criticality (Taipale et al. 2006a).

The first interview round was connected with a larger survey. For the survey and the first interview round, the selection for the sample from the population of 62 organizations was based on probability sampling. The original population of OUs was identified with the help of national and local authorities. The OUs were in a random order in our database, and every second OU was selected. The sample of the survey consisted of 30 OUs. Results of the survey were used in selection of the case OUs. From this sample of 30 OUs, 5 OUs were further selected as case OUs for the second, third and fourth interview rounds. Now the sampling was theoretical (Pare and Elam 1997), where

Table 1 OUs and interviewees

Interview round(s)	OU	Business	Company size	Interviewees
1st	All 26 OUs including cases from A to E	Automation or telecommunication domain	The OUs were parts of large companies (53%) and small and medium-sized enterprises (47%)	Managers, 52% of the interviewees were responsible for both development and testing, 28% were responsible for testing, and 20% were responsible for development.
2nd to 4th	Case A	Manufacturing execution systems (MES) producer and integrator	Large/international	Testing manager, tester, developer
2nd to 4th	Case B	Software producer and testing service provider	Small/national	Testing manager, tester, developer
2nd to 4th	Case C	Process automation and information management provider	Large/international	Testing manager, tester, developer
2nd to 4th	Case D	Electronics manufacturer	Large/international	Testing manager, 2 testers, developer
2nd to 4th	Case E	Testing service provider	Small/national	Testing manager, tester, developer

the researchers' goal was not to collect a representative sample of all possible variations, but to gain a deeper understanding of the analyzed cases. In theoretical sampling, we selected polar types (Eisenhardt 1989), which mean that the cases represented different types of OUs highlighting diversity, for example a different business orientation, different size, or domestic versus international operation.

3.2 Data collection

Our data collection consisted of four interview rounds, presented in Table 1. The company size classification has been taken from the classification by the European Union (2003). During the first interview round, managers responsible for testing and/or development were interviewed. We selected managers as the first interviewees because they have a wide experience, and they are able to give an overview of testing in the OU. The purpose of the first interview round was to collect survey data and answers to open questions, as well as give information for the theoretical sampling, i.e. for the selection of the case OUs. The interviews were tape-recorded for further qualitative analysis. The results of the survey conducted during the first interview round are presented in our previous publications (Taipale et al. 2006a; Taipale et al. 2006b; Taipale et al. 2006c).

The second, third, and fourth interview rounds were conducted in the selected case OUs. The purpose of these interview rounds was to gain detailed understanding of the practice of software testing in the OUs.

The interviewees of the second round were managers of testing, those of the third round were testers, and those of the fourth round were developers. The new ideas from each round were reflected in the themes of the following interview rounds. The data collection process of all interviews generated a transcription of 946 pages.

3.3 Data analysis

The grounded theory method was used in analyzing the interview data from the case OUs. According to Strauss and Corbin (1990), grounded theory contains three data analysis steps: open coding, where categories of the study are extracted from the data; axial coding, where connections between the categories are identified; and selective coding, where the core category is identified and described. A category combines different variations of the same phenomenon and it is given a conceptual label. Categorizing is used to reduce the number of units to work with (Strauss and Corbin 1990).

In practice, the data collection and analysis overlapped and merged because the process proceeded iteratively. The interviews were transcribed and analyzed by four researchers after each interview round to collect new issues, and to see if it was worthwhile to continue the data collection procedure. The general rule in grounded theory is to sample until theoretical saturation is reached, which means until (1) no new or relevant data seem to emerge regarding a category; (2) the category development is dense, insofar as all the paradigm elements are accounted for, along with variation and process; (3) the relationships between the categories are well established and validated (Strauss and Corbin 1990). Theoretical saturation was reached during the fourth interview round, as new categories no longer appeared, categories were not merged, shared, or removed, the attributes or attribute values of the categories did not change, and the relationships between

the categories were considered stable, i.e. already described phenomena began to be repeated in the data.

The objective of open coding was to classify the data into categories and identify leads in the data. The open coding of the interviews was done with ATLAS.ti software (Atlas 2009). An example of open coding is given in Table 2. The process started with "seed categories" (Miles and Huberman 1994) that contained essential stakeholders (such as organizational units and roles), phenomena (such as testing knowledge, test automation, and reuse), and problems. Seaman (1999) notes that the initial set of codes (seed categories) comes from the goals of the study, the research questions, and predefined variables of interest. In our case, the seed categories were deduced by the research group from the research question, from areas of interest defined in the earlier phases of the research project (Taipale et al. 2005), and from phenomena observed when transcribing the interviews in the first round. These identified seed categories were used as a starting point in the open coding, where all the interview material was coded. The phase produced a large set of other codes and categories in addition to the seeds. Four researchers participated in the data analysis process.

When the open coding and the data collection were considered completed, three researchers were each given a viewpoint (test automation, knowledge management, and processes) from which they started the axial coding of the data. The objective of axial coding was to develop categories, dimensions, and causal conditions or any kinds of connections between the categories further. However, during the axial coding, changes were made to the existing categories so that they would be more descriptive and make further analysis easier. The categories were developed further by defining the dimensions. The dimensions represent the locations of the property or the attribute of a category along a continuum (Strauss and Corbin 1990).

The objective of selective coding was to identify the core category (Strauss and Corbin 1990) and to relate it systematically to the other categories. We found that there was no single core category among the existing categories. According to Strauss and Corbin (1990), sometimes the core category is one of the existing categories, and at other times no single category is broad enough to cover the central phenomenon. In that case, the central phenomenon must be given a name. In this study, the creation of the core category meant the identification of the affecting factors (categories) that explain the practice of software test automation, and finding the relationships between these categories.

4 Analysis results

As suggested by Eisenhardt (1989), we collected evidence for each category iteratively, explored the logic across the case OUs, and searched for evidence from the data for each identified relationship between the categories. In open coding we concentrated on categories that were in some way related to software test automation. The analysis continued by merging categories that seemed to be closely connected and by dividing categories that under a close inspection seemed to contain separate phenomena. In addition, we defined dimensions for the categories. An example of merging categories is given in Table 3.

The purpose was to get a comprehensive picture of the current state of test automation in the case OUs. After that, we analyzed the data to find categories that influence the use of test automation. The process was repeated until we could explain their influence on test automation in the case OUs. During this phase all the categories from open coding, including seeds and freshly identified ones, were further developed to the final categories described below (Table 4).

The category *Use of Test Automation* describes how widely test automation is utilized in the OU. This is the core category of our study.

The category *Benefits of Test Automation* describes the observed features and characteristics of test automation that makes it beneficial.

The category *Disadvantages of Test Automation* describes the observed negative features and characteristics of test automation.

The category *Development of Test Automation* describes the observations and opinions of how test automation is being developed or should be developed in the OU.

The category *Type of Tested Products* describes what kind of products are developed and tested in the OU. Sommerville (1995) divides software products into two

Table 2 Example of open coding using ATLAS.ti software

Interview transcript	Categories and observations in open coding. Category: Code
"It is always expensive to set up an automated system. The price may be tenfold compared to one test.—But later, if there is more repetition, the unit cost per test decreases quite significantly at some point."	Advantages and Disadvantages of test automation: Price of the system Reuse: Repetition decreases unit cost

Table 3 Example of merging categories and observations

Seed category: Observation	New merged category: Observation
Testing know-how*:* Issues of testing know-how including explicit (codified) and tacit (personalized) know-how	Testing knowledge: Issues of knowledge management strategy in testing
Communication and interaction*:* Issues of knowledge transfer in testing	

Table 4 Final categories for the case OUs

Category	Dimension	Description
Use of test automation	Low–Wide	The current level of test automation utilization in the OU
Benefits of test automation	List of benefits	The general benefits of test automation according to the interviewees
Disadvantages of test automation	List of disadvantages	The general disadvantages of test automation observed by the interviewees
Development of test automation	Development ideas	How testing will/should be developed in the OU according to the interviewees
Type of tested products	Generic–customized	The type of software tested in the OU
Customer	Description and a distinction between internal versus external customers	Description of the OU's customers and their business and systems
Testing knowledge	Type of knowledge (e.g. domain knowledge)	The type of knowledge needed in performing the testing tasks in the OU
Test level	Unit–System	The level of testing performed in the OU

broad classes, generic products and customized products. Generic products are produced by a development organization and sold on an open market to any customer who is able to buy them. Organizations which develop generic products for open markets are then called product-oriented organizations. On the other hand, customized products are systems commissioned by a particular customer and developed specially for that customer by some contractor. These organizations develop their software to a small, limited number of customers. Complexity increases if the software product has many interfaces to third party systems.

The category *Customer* describes the customer base of the OU. Customers may vary in number and also their (business) type. Customers can be either internal or external.

The category *Testing Knowledge* describes the type of knowledge needed in performing testing tasks, especially the amount of domain knowledge needed in performing testing tasks in the OU.

Test Level describes the level on which testing is performed in the OU, such as unit, integration or systems testing.

4.1 Description of cases

4.1.1 Case A: a MES producer and integrator

Case A develops and tests manufacturing execution systems (MES). Its services include system integration and customizing of systems. The testing tasks conducted in Case A are mostly systems testing and integration testing. Their customized systems are based on a uniform product kernel, which is customized according to the needs of the customers. In managing the product kernel and its variations, internally developed tools are used.

Because different customers have different technical infrastructures, managing all the different variations of technologies is considered challenging. The OU has to take into account possible version variations of, for example, operating systems, new interfaces, and other systems that different customers have. According to the interviewees, their systems become more and more complex, and integrating systems delivered by multiple parties is sometimes more challenging than the customer had anticipated.

Test automation is utilized in the development of product development tools, but not widely in the final customized system. There are plans to develop test automation further so that there would be automation in testing the product kernel. According to the interviewees, the lack of resources prevents the development of test automation. It is difficult to get additional funding and personnel for test automation development. Also the pressure to stay in the delivery schedule ensures that there is no time for the existing personnel to develop test automation.

One of the major problems in systems testing is that the MES systems can only be fully tested at the external

customer's site because they have interfaces to customer-specific systems, which can not be fully simulated in the development environment. According to the interviewees, there might be potential for test automation in integration testing where the test cases are repeatable, but it is not yet implemented due to lack of time, personnel, and funding. The system testing of MES systems is mainly manual, and the testers need wide domain knowledge.

> "It (domain knowledge) is essential because the tester has to be in the end-user's position." (Case A, testing manager).

4.1.2 Case B: software producer and testing service provider

Case B develops and tests its own software products and offers testing services to external customers. The tested products are standardized and generic, both in testing services and in its own software production. The Case B organization follows test plans provided by their customer, and the testing tasks are mainly conducted on the systems testing level.

Although Case B has not made investments in test automation regarding their testing services to external customers, their testing services include working with the customer's highly automated testing systems. The testing services consist of scripting test cases for the customer's testing systems, supervising the execution of systems testing, and reporting the test results.

There is a project under way in the OU to introduce systematic unit testing to the development of their own software products. Faster development and testing is the main motivation for developing unit test automation.

> "There is often less time for testing and we should test more, and by using automation it can be made possible." (Case B, developer).

The testing manager feels that test automation makes testing faster, but also notes that there always has to be some human involvement in testing. For example, they have problems with a customer's automated testing systems that do not work properly. There have been problems for example in test cases, test scripts, and the testing environment. Also, according to an interviewed developer, in automated unit testing there has to be some human reasoning in the selection of test cases to achieve good test coverage and to evaluate testing systems. This also changes how testing resources are used, as illustrated by the following quotation.

> "We have to test the testing systems constantly [because of changes]." (Case B, testing manager).

4.1.3 Case C: process automation and information management provider

Case C tests and develops customized process automation and information management systems. The testing in Case C consists mostly of systems testing. The systems are large and complex, and depend on other suppliers' systems. Especially interoperability testing is found difficult, because it can often be done only at the external customer's site.

Managing several versions of the product for different platforms and operating systems is considered challenging. Interoperability and changes in the technological infrastructure hinder test automation, because part of the test environment is dedicated to a certain platform, and changes in the technological environment may make the part of the test environment obsolete.

There seems to be limited utilization of test automation, especially in the systems testing phase. According to the tester, there have been attempts to develop test automation, but the results have not been satisfactory. At some point there has been a student working on his thesis, who wrote automation scripts that made the testers work easier. The testers used them for a while, but after the student left the company, there was nobody to maintain the system.

> "We have been trying to include automation in our operations for years. There have been projects to automate some testing phases, but they have not been successful. It's too difficult. And we have also found out that it doesn't suit our type of products." (Case C, tester)

One issue that may hinder the development of test automation is that the employees may have a negative attitude towards it, or they may resist change. For example, some employees prefer manual testing because they feel that operating and maintaining the test automation system takes more work than manual testing. Also learning new skills while concurrently keeping the project in schedule is seen as too challenging. Lack of resources is indicated as one of the reasons for limited application of test automation. It is seen that developing and maintaining a test automation system requires a considerable amount of time, money, and human resources.

The interviewees work mainly in the system testing area (except for the interviewed developer). It seems that manual testing is the primary strategy in systems testing because the testers are required to have a significant amount of domain knowledge when they put themselves in the end-user's position. Especially automating user interface testing is considered difficult, and manual testing is preferred.

In the testing of customized systems, tacit domain knowledge is seen essential, because testing is mainly

systems testing and the testers simulate end-user operations. This end-user simulation emphasizes the personalization-based knowledge management strategy.

4.1.4 Case D: electronics manufacturer

Case D develops and tests generic, independent, and highly standardized products. The quality of their end-products is of high importance because they must protect their brand name and the costs of recalls in mass production would be too big. Testing in Case D consists mostly of systems testing and measurements of the products developed inside the organization.

In Case D, test automation is widely utilized and constantly developed, and it is seen that it improves the quality of testing. According to observations, using test automation does not necessarily shorten the product development project, but it allows running more tests than doing it manually. The wider testing improves quality through better test coverage.

However, Case D has problems in the development of test automation. One of the problems is that they have few specifications against which they can test. It is also noted that test automation requires significant investments. Case D is the only OU in our study (including the survey) where the automation costs are higher than the personnel costs. The greatest problem of Case D in test automation is that sometimes the test automation systems are faulty themselves and need maintenance.

Case D uses a quality system, which is applied throughout the organization. Reuse is taken into account in product development and testing infrastructure development.

Because there is less need for manual testing in Case D, there is also less need for the testers' domain knowledge. Knowledge of testing tools and methods is regarded as more important than knowledge of the application domain.

4.1.5 Case E: testing service provider

Case E provides testing services to other organizations. Case E does not utilize test automation because they have so many different customers operating in different domains.

> "We do not use test automation." (Case E, tester & developer)

Although the tested products are mainly generic, they vary so much from customer to customer that it is difficult to find enough similarities between them to build test automation systems which could be applied to all the customers' systems. In other words, there are little chances for reuse between the different customer projects. It is seen that it is not profitable to build test automation systems that can be used only once, because the costs of setting up a test automation infrastructure are significant.

> "It is always expensive to set up an automated system. The price may be tenfold compared to one test.—But later, if there is more repetition, the unit cost per test decreases quite significantly at some point." (Case E, testing manager)

Testing customized systems is not preferred in Case E. According to the interviewees, testing customized systems is problematic because of their complexity, which makes effort estimation difficult.

In addition to test execution, the testing tasks in Case E consist of planning of testing, documentation, and defining the testable components and systems. Case E's customer base fully consists of external customers. Knowledge of customers' software development processes is seen as more important than domain knowledge.

4.2 Observations

There were both clear differences and similarities between the cases in their approaches to software testing and test automation. Case D was the OU where test automation was widely utilized, developed, and systematically taken into account throughout the software development process. Cases A and C were similar in a sense that they had no systematic approach to systems test automation. Complex systems and the need for domain knowledge seemed to be the most decisive factors in hindering test automation investments, especially on the systems testing level. It was also observed in Case C that there may have been some resistance among the testers in accepting test automation. Case B was ambivalent in terms of test automation. Although they had not themselves invested in test automation regarding their testing service business, they worked with their customers' automated testing systems. Automated unit testing had also been introduced to their own internal product development. With the testing service provider E, no significant utilization of test automation was observed. There was no reuse value for software test automation because the projects were short, and different customers have very different domains, although the products were mainly generic. The investment in test automation infrastructure was seen to be too expensive for basically one-time use.

In the following, we summarize our findings in the form of four observations.

4.2.1 *Observation 1: "Automated software testing makes testing faster, improves quality and facilitates reuse, but causes new costs"*

The benefits and disadvantages of test automation observed by the interviewees are presented in Table 5. The major benefits of test automation seem to consist of quality improvement through better test coverage, the fact that more testing can be done in less time and fluent reuse of testware.

The major disadvantages are the costs. The cost items mentioned in the interviews consisted of direct investment costs, extra implementation time, and the need for extra human resources. The implementation of the test automation infrastructure was viewed as costly and time-consuming. It was also seen that test automation requires maintenance, which should be taken into account when evaluating its cost effectiveness. The training of testers and developers to use the test automation systems should also be taken into account. Unreliability of test automation systems was also mentioned as a problem. Automated testing systems consist of hardware and software and suffer from the same stability and error issues as any other systems.

4.2.2 *Observation 2: "The properties of the tested products affect the applicability of test automation"*

If the tested products are generic and independent of third-party systems, test cases and therefore test automation needs are easier to specify. Customized systems have more variability and interfaces to third-party systems, and therefore it is more difficult to carry out test specifications as the system environment and functionalities vary between customers.

If the testing organization (such as Case E) tests several different types of products for external customers, there are few chances for finding similarities between them and automating testing tasks. On the other hand, products based on a uniform product kernel or variants of a product family (software product line architecture) together with internal customers may enable the use of test automation, as in Case D.

Table 5 Observed benefits and disadvantages of testing automation

Benefits	Disadvantages
Quality improvement	Implementation costs
More testing in less time	Maintenance costs
Reuse	Training costs
	Unreliability

4.2.3 *Observation 3: "Test automation does not eliminate personnel costs"*

The need for human involvement in supervising, maintaining, and developing test automation was often emphasized in the data. Test automation seemed to be difficult to implement if domain knowledge was crucial in the testing tasks. In addition, automated testing systems have to be supervised in the case of faults. The need for domain knowledge was especially observed in the systems testing phase of customized systems. On the other hand, knowledge of testing tools and methods was emphasized in connection with generic, independent, and highly standardized products—and high level testing automation. An important impact of test automation is that it frees up resources for manual testing by automating repetitive tasks. However, it does not replace manual testing, which is needed for all tasks that cannot be standardized and repeated.

4.2.4 *Observation 4: "Testers' attitudes and habits affect the utilization of test automation"*

Motivating the employees to adopt new testing practices seems to be an issue, but resistance against changes is also a universal phenomenon. Adopting test automation means especially that the testers' way of working changes and they would probably need training to use automated testing systems. The testers' time and resources to learn new skills are limited, because their effort is directed to keep the project in schedule as well as within the budget.

4.3 Towards a better understanding of the practice of test automation

In grounded theory, the core category gives an inductive theoretical explanation to the phenomenon under study. In this study, we iteratively collected evidence for each identified category, and explored and interpreted the practices of the case OUs to form theoretical understanding. We chose to explain the core category with two stereotypes highlighting the factors that affect the use of test automation. With the understanding yielded by the analysis, we described two stereotypes of organizations. We collected observations about facilitating or hindering the use of test automation and associated them with the stereotypes, presented in Table 6.

Product-oriented organizations that develop generic and independent software have better possibilities to utilize test automation than service-oriented organizations producing customized software. The need to adapt rapidly to varying external customers' needs and technical environments seems to be common for organizations producing

Table 6 Stereotypes of organizations in test automation

Features in organizations	Features that facilitate test automation	Features that hinder test automation
Product type	Generic, independent and similar products	Customized, complex and dissimilar products
Need for human involvement	Low	High
Changes in underlying technology	Standardized infrastructure	Rapid changes
Customer base	Internal customers	External customers

customized software. Service-oriented organizations producing customized software need to develop abilities to adapt to rapid changes in the environment and underlying technology, and therefore, the costs related to the implementation of test automation may not always be covered by the benefits.

5 Discussion

The objective of this case study was to observe and identify factors that affect the use of test automation in different types of organizations. Our cases included three different types of organizations: product-oriented software development, customized systems development, and testing service providers. We interviewed employees in different organizational positions in each of the cases. The interview data was analyzed using the grounded theory method.

5.1 Comparison of results to earlier research

The major perceived benefits of test automation include quality improvement through a better test coverage, the fact that more testing can be done in less time with fluent reuse of testware. However, an observation was made that to achieve a better test coverage, automation alone is not enough, but human involvement is needed in the selection of test cases.

We found that the main disadvantages of test automation are the costs, which include implementation , maintenance , and training costs. Implementation costs include direct investment costs, time, and human resources. According to Fewster (2001), there is a connection between implementation and maintenance costs. If the test automation system is designed with the minimization of maintenance costs in mind, the implementation costs increase, and vice versa. Fewster (2001) views that if the maintenance of test automation is ignored, updating an entire automated test suite can cost as much, or even more than the cost of performing all the tests manually. Training costs rise because employees have to change their working practices. Especially testers' daily tasks may be transformed from manual testing to maintaining automated testing systems. This change requires major retraining of the testing staff. For example, Bach (1997) has noticed that "all automated test suites require human intervention, if only to diagnose the results and fix broken tests", a task that requires the staff to learn to use and understand the automation system.

According to our observations, the properties of the tested products influence the use of test automation. If the tested products are generic and independent, automated tests are easier to specify. If the tested products are customized and complex, specifying automated tests seems to become more expensive and difficult to implement feasibly. Test automation systems developed for customized systems may have limitations in reusability, which is important factor for financially feasible test automation. However, it is possible for customized systems and their testing systems as well to have a long life span and a high degree of reuse, as may be the case with e.g. homegrown ERP systems.

The reusability of a test automation system is essential in ensuring profitable investment. Jones (1994) views that reuse can be successful when the reusable materials are of a high quality, i.e. "certified to levels of quality that approach or achieve zero defects" and when artifacts are constructed so that subsequent reuse is straightforward and efficient. According to Jones (1994), other barriers to reuse are finding the time and funds to construct reusable materials in the schedule and the cost pressure under which most software projects are. According to our observations, the testing schedules were tight, and when the customer defined the schedule and budget, there were no extra resources left for developing software test automation.

If there is a great need for domain knowledge in testing, automating testing tasks becomes difficult. Domain knowledge is often tacit and embedded in employees, and transferring tacit knowledge is difficult. According to Bertolino (2007), one of the dreams of software testing is 100% automatic testing. However, according to our, as well as for example Bach's (1997) observations, this objective may be beyond realistic expectations.

5.2 Validity of the study

The purpose of this qualitative study was to understand and describe the specific phenomena related to test automation in five case OUs. This kind of an effort requires

interpretation and exploration, and therefore the main instrument of research is the researcher. Robson (2002) lists three threats to validity in this kind of research: reactivity (the interference of the researcher's presence), researcher bias, and respondent bias, and strategies that reduce these threats. We have used these strategies in the following way: the research involvement was prolonged, as the research lasted for more than 3 years and consisted of several phases and data collection rounds. Audit trail was used: all interviews were recorded and transcribed. The notes and memos of the study have been preserved, and the preliminary data coding and analysis results are available through the analysis tool used, ATLAS.ti. Three types of triangulation presented by Denzin (1978) were used: observer, methodological, and theory triangulation.

The strongest method for ensuring the overall validity of the study was the triangulation. To reduce the bias caused by researchers, we used observer triangulation. The bias caused by the method was minimized using methodological triangulation. Earlier publications concerning quantitative and qualitative studies (Taipale et al. 2006c; Taipale et al. 2006a; Taipale et al. 2005; Taipale et al. 2006b; Taipale et al. 2007; Karhu et al. 2007; Karhu et al. 2009) have approached software testing from different viewpoints, and therefore they enforce theory triangulation. Methodological triangulation means that multiple research methods are used and their results are compared to each other. In this study, methodological triangulation consisted of comparing the results of this study with the results of earlier studies.

In observer triangulation, researchers with different backgrounds and experiences study the same research topic and participate in the data collection. In this study, the analysis was carried out by four researchers, whose interpretations completed each other, and therefore made the study more trustworthy. The obvious limitation of the study is the number of case OUs. Increasing the number of cases to cover a wider variety of different types of organizations could reveal more details, but we believe that this study has already revealed important empirical observations on software test automation.

6 Conclusions

Our objective was to examine and analyze test automation systems in practice, and to discuss the impact that test automation has in the industry. The rationale for our study was that the software industry loses vast sums of money annually due to inadequate testing infrastructures (Tassey 2002). Furthermore, the testing process itself is in many cases the largest investment in the software project (Kit 1995).

It seems that developing test automation could be a solution with several positive effects. Automated testing yields benefits for example in automated test case generation and execution, leaving more human resources to test new features or to do explorative testing and decreasing costs by relieving the manual workload. However, the application of test automation is not as straightforward as it seems, as there are several pitfalls that need to be considered (Persson and Yilmaztürk 2004; Kaner 2000).

Our study focused on five software testing OUs to analyze the applicability of test automation systems in different software business domains. Our findings suggest that the properties of the tested products affect the applicability of test automation, test automation does not eliminate personnel costs, and the testers' attitudes and habits affect the utilization of test automation. It seems that the optimal case for automated software testing would be a standardized product with a stable, consistent platform and cases that yield unambiguous results which can be verified with minimal human intervention. In addition, as software automation requires effort to maintain and develop, the system should also be easily reusable in different software projects to have feasible return for the initial investment. Hardware and software changes are among the greatest hindrances to a successful implementation of a test automation infrastructure.

According to our results, it seems that organizations which are considering test automation should first consider if their test process has activities which are feasible to automate, as there are several requirements for successful test automation. If applicable, test automation can be used to supplement the test process and offer quality control tools for testing. However, there are still several practical issues, such as the application area selection or availability of suitable tools, which should be taken into account when implementing test automation. Suitable standards and practices, application areas, delivery models (such as testing in the cloud), methods, and tools are important future research topics in the practice of test automation.

Acknowledgment This study was supported by the ESPA-project (http://www.soberit.hut.fi/espa), funded by the Finnish Funding Agency for Technology and Innovation, and by the companies mentioned in the project pages.

References

Atlas (2009) ATLAS.ti—The knowledge workbench. Scientific Software Development, http://www.atlasti.com Acessed 29 July 2009

Bach J (1997) Test automation snake oil. In: Proceedings of 14th international conference and exposition on testing computer

Berner S, Weber R, Keller RK (2005) Observations and lessons learned from automated testing. In: Proceedings of the 27th

international conference on Software engineering, St. Louis, Mo, USA, 571–579. doi:10.1145/1062455.1062556
Bertolino A (2007) Software testing research: achievements, challenges, dreams, future of software engineering. IEEE Computer Society, pp 85–103. doi: 10.1109/FOSE.2007.25
Blackburn M, Busser R, Nauman A (2004) Why model-based test automation is different and what you should know to get started.In: International conference on practical software quality and testing
Briand LC (2007) A critical analysis of empirical research in software testing. In: First international symposium on empirical software engineering and measurement Madrid, Spain: IEEE Computer Society. doi: 10.1109/ESEM.2007.40
Carter CR, Dresner M (2001) Purchasing's role in environmental management: Cross-functional development of grounded theory. J Supply Chain Manag 37:12–28
Denzin NK (1978) The research act: a theoretical introduction to sociological methods. McGraw Hill, New York
Dustin E, Rashka J, Paul J (1999) Automated software testing: introduction management, and performance. Addison-Wesley, Boston
Eisenhardt KM (1989) Building theories from case study research. Acad Manag Rev 14:532–550
EU (2003) SME Definition. European Commission
Fewster M (2001) Common mistakes in test automation.In: Fall test automation conference, Boston
Glaser B, Strauss AL (1967) The discovery of grounded theory: strategies for qualitative research. Aldine, Chicago
Hartman A, Katara M, Paradkar A (2007) Domain specific approaches to software test automation. In: Proceedings of 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, Dubrovnik, 621–622, 2007. doi: 10.1145/1287624.1287730
ISO/IEC (2002) ISO/IEC 15504-1 Information Technology—Process Assessment—Part 1: Concepts and Vocabulary
Jones C (1994) Economics of software reuse. IEEE Computer 27:106–107. doi:10.1109/2.299437
Kaner C (1997) Improving the maintainability of automated test suites. Softw QA 4(4):28
Kaner C (2000) Architectures of test automation. In: Software testing, analysis and review conference (Star) West, San Jose
Karhu K, Taipale O, Smolander K (2007) Outsourcing and knowledge management in software testing. In: Proceedings of the 11th international conference on evaluation and assessment in software engineering (EASE), Keele University, Staffordshire, UK, BCS, 2–3 April 2007
Karhu K, Repo T, Taipale O, Smolander, K (2009) Empirical observations on software testing automation. In: Proceedings of the 2nd international conference on software testing, verification and validation, Denver, CO, USA. doi: 10.1109/ICST.2009.16
Kit E (1995) Software testing in the real world: improving the process. Reading. Addison-Wesley, Boston
Klein HK, Myers MD (1999) A set of principles for conducting and evaluating interpretive field studies in information systems. MIS Q 23:67–94. doi:10.2307/249410
Mantere T (2003) Automatic software testing by genetic algorithms. Dissertation, Universitas Wasaensis, Vaasa, vol. 112
Miles MB, Huberman AM (1994) Qualitative data analysis. SAGE Publications, Thousand Oaks
Ng SP, Murmane T, Reed K, Grant D, Chen TY (2004) A preliminary survey on software testing practices in Australia. In: Proceedings of 2004 Australian software engineering conference, Melbourne, Australia, 116–125
Pare′ G, Elam JJ (1997) Using case study research to build theories of IT implementation.In: IFIP TC8 WG international conference on information systems and qualitative research, Philadelphia, USA
Persson C, Yilmaztürk N (2004) Establishment of automated regression testing at ABB: industrial experience report on 'Avoiding the Pitfalls'.In: the 19th international conference on automated software engineering (ASE'04): IEEE Computer Society. doi: 10.1109/ASE.2004.1342729
Ramler R, Wolfmaier K (2006) Economic perspectives in test automation: balancing automated and manual testing with opportunity cost. AST'06 Shanghai. ACM, China. doi: 10.1145/1138929.1138946
Robson C (2002) Real world research, 2nd edn. Blackwell Publishing, Oxford
Santos-Neto P, Resende R, Pádua C (2007) Requirements for information systems model-based testing. In: Proceedings of the 2007 ACM symposium on applied computing, Seoul, Korea, 1409–1415. doi: 10.1145/1244002.1244306
Seaman CB (1999) Qualitative methods in empirical studies of software engineering. IEEE Trans Softw Eng 25:557–572. doi: 10.1109/32.799955
Shaw M (2003) Writing good software engineering research papers. In: Proceedings of the 25th international conference on software engineering (ICSE' 03), Portland, pp 726–736
Sjöberg DIK, Dybå T, Jorgensen M (2007) The future of empirical methods in software engineering research. FOSE'07: 2007 Future of Software Engineering, pp. 358–378. doi: 10.1109/FOSE.2007.30
Smolander K, Rossi M, Purao S (2008) Software architectures: blueprint literature, language or decision. Eur J Inf Syst 17: 575–588
Sommerville I (1995) Software engineering. Addison Wesley, Essex
Strauss A, Corbin J (1990) Basics of qualitative research: grounded theory procedures and techniques. SAGE Publications, Newbury Park
Taipale O, Smolander K, Kälviäinen H (2005) Finding and ranking research directions for software testing. In: European software process improvement and innovation conference, Budapest, Hungary, pp 39–48. doi: 10.1145/336512.336532
Taipale O, Smolander K, Kälviäinen H (2006a) A survey on software testing.In: The 6th international SPICE conference on software process improvement and capability dEtermination (SPICE' 2006), Luxembourg
Taipale O, Smolander K, Kälviäinen H (2006b) Factors affecting software testing time schedule. In: Australian software engineering conference, Sydney. doi: 10.1109/ASWEC.2006.27
Taipale O, Smolander K, Kälviäinen H (2006c) Cost reduction and quality improvement in software testing.In: Software quality management conference, Southampton, UK
Taipale O, Karhu K, Smolander K (2007) Observing software testing practice from the viewpoint of organizations and knowledge management. In: Proceedings of the 1st international symposium on empirical software engineering and measurement (ESEM), 2007, Madrid, Spain, IEEE. doi: 10.1109/ESEM.2007.18
Tassey G (2002) The economic impacts of inadequate infrastructure for software testing. US national institute of standards and technology report, RTI Project Number 7007.011
Torkar R, Mankefors S (2003) A survey on testing and reuse. In: IEEE international conference on software-science, technology and engineering (SwSTE'03), Herzlia, Israel