Hindawi Publishing Corporation
Advances in Software Engineering
Volume 2010, Article ID 620836, 18 pages
doi:10.1155/2010/620836

Research Article

Software Test Automation in Practice: Empirical Observations

Jussi Kasurinen, Ossi Taipale, and Kari Smolander

Department of Information Technology, Laboratory of Software Engineering, Lappeenranta University of Technology, P.O. Box 20, 53851 Lappeenranta, Finland

Correspondence should be addressed to Jussi Kasurinen, jussi.kasurinen@lut.fi

Received 10 June 2009; Revised 28 August 2009; Accepted 5 November 2009

Academic Editor: Phillip Laplante

The objective of this industry study is to shed light on the current situation and improvement needs in software test automation. To this end, 55 industry specialists from 31 organizational units were interviewed. In parallel with the survey, a qualitative study was conducted in 12 selected software development organizations. The results indicated that the software testing processes usually follow systematic methods to a large degree, and have only little immediate or critical requirements for resources. Based on the results, the testing processes have approximately three fourths of the resources they need, and have access to a limited, but usually sufficient, group of testing tools. As for the test automation, the situation is not as straightforward: based on our study, the applicability of test automation is still limited and its adaptation to testing contains practical difficulties in usability. In this study, we analyze and discuss these limitations and difficulties.

1. Introduction

Testing is perhaps the most expensive task of a software project. In one estimate, the testing phase took over 50% of the project resources [1]. Besides causing immediate costs, testing is also importantly related to costs related to poor quality, as malfunctioning programs and errors cause large additional expenses to software producers [1, 2]. In one estimate [2], software producers in United States lose annually 21.2 billion dollars because of inadequate testing and errors found by their customers. By adding the expenses caused by errors to software users, the estimate rises to 59.5 billion dollars, of which 22.2 billion could be saved by making investments on testing infrastructure [2]. Therefore improving the quality of software and effectiveness of the testing process can be seen as an effective way to reduce software costs in the long run, both for software developers and users.

One solution for improving the effectiveness of software testing has been applying automation to parts of the testing work. In this approach, testers can focus on critical software features or more complex cases, leaving repetitive tasks to the test automation system. This way it may be possible to use human resources more efficiently, which consequently may contribute to more comprehensive testing or savings in the testing process and overall development budget [3]. As personnel costs and time limitations are significant restraints of the testing processes [4, 5], it also seems like a sound investment to develop test automation to get larger coverage with same or even smaller number of testing personnel. Based on market estimates, software companies worldwide invested 931 million dollars in automated software testing tools in 1999, with an estimate of at least 2.6 billion dollars in 2004 [6]. Based on these figures, it seems that the application of test automation is perceived as an important factor of the test process development by the software industry.

The testing work can be divided into manual testing and automated testing. Automation is usually applied to running repetitive tasks such as unit testing or regression testing, where test cases are executed every time changes are made [7]. Typical tasks of test automation systems include development and execution of test scripts and verification of test results. In contrast to manual testing, automated testing is not suitable for tasks in which there is little repetition [8], such as explorative testing or late development verification

tests. For these activities manual testing is more suitable, as building automation is an extensive task and feasible only if the case is repeated several times [7, 8]. However, the division between automated and manual testing is not as straightforward in practice as it seems; a large concern is also the testability of the software [9], because every piece of code can be made poorly enough to be impossible to test it reliably, therefore making it ineligible for automation.

Software engineering research has two key objectives: the reduction of costs and the improvement of the quality of products [10]. As software testing represents a significant part of quality costs, the successful introduction of test automation infrastructure has a possibility to combine these two objectives, and to overall improve the software testing processes. In a similar prospect, the improvements of the software testing processes are also at the focus point of the new software testing standard ISO 29119 [11]. The objective of the standard is to offer a company-level model for the test processes, offering control, enhancement and follow-up methods for testing organizations to develop and streamline the overall process.

In our prior research project [4, 5, 12–14], experts from industry and research institutes prioritized issues of software testing using the Delphi method [15]. The experts concluded that process improvement, test automation with testing tools, and the standardization of testing are the most prominent issues in concurrent cost reduction and quality improvement. Furthermore, the focused study on test automation [4] revealed several test automation enablers and disablers which are further elaborated in this study. Our objective is to observe software test automation in practice, and further discuss the applicability, usability and maintainability issues found in our prior research. The general software testing concepts are also observed from the viewpoint of the ISO 29119 model, analysing the test process factors that create the testing strategy in organizations. The approach to achieve these objectives is twofold. First, we wish to explore the software testing practices the organizations are applying and clarify the current status of test automation in the software industry. Secondly, our objective is to identify improvement needs and suggest improvements for the development of software testing and test automation in practice. By understanding these needs, we wish to give both researchers and industry practitioners an insight into tackling the most hindering issues while providing solutions and proposals for software testing and automation improvements.

The study is purely empirical and based on observations from practitioner interviews. The interviewees of this study were selected from companies producing software products and applications at an advanced technical level. The study included three rounds of interviews and a questionnaire, which was filled during the second interview round. We personally visited 31 companies and carried out 55 structured or semistructured interviews which were tape-recorded for further analysis. The sample selection aimed to represent different polar points of the software industry; the selection criteria were based on concepts such as operating environments, product and application characteristics (e.g., criticality of the products and applications, real time operation), operating domain and customer base.

The paper is structured as follows. First, in Section 2 we introduce comparable surveys and related research. Secondly, the research process and the qualitative and quantitative research methods are described in Section 3. Then the survey results are presented in Section 4 and the interview results are presented in Section 5. Finally, the results and observations and their validity are discussed in Section 6 and closing conclusions are discussed in Section 7.

2. Related Research

Besides our prior industry-wide research in testing [4, 5, 12–14], software testing practices and test process improvement have also been studied by others, like Ng et al. [16] in Australia. Their study 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 barrier to developing testing was the lack of expertise in adopting new testing methods and the costs associated with testing tools. In their study, only 11 organizations reported that they met testing budget estimates, while 27 organizations spent 1.5 times the estimated cost in testing, and 10 organizations even reported a ratio of 2 or above. In a similar vein, Torkar and Mankefors [17] surveyed different types of communities and organizations. They found that 60% of the developers claimed that verification and validation were the first to be neglected in cases of resource shortages during a project, meaning that even if the testing is important part of the project, it usually is also the first part of the project where cutbacks and downscaling are applied.

As for the industry studies, a similar study approach has previously been used in other areas of software engineering. For example, Ferreira and Cohen [18] completed a technically similar study in South Africa, although their study focused on the application of agile development and stakeholder satisfaction. Similarly, Li et al. [19] conducted research on the COTS-based software development process in Norway, and Chen et al. [20] studied the application of open source components in software development in China. Overall, case studies covering entire industry sectors are not particularly uncommon [21, 22]. In the context of test automation, there are several studies and reports in test automation practices (such as [23–26]). However, there seems to be a lack of studies that investigate and compare the practice of software testing automation in different kinds of software development organizations.

In the process of testing software for errors, testing work can be roughly divided into manual and automated testing, which both have individual strengths and weaknesses. For example, Ramler and Wolfmaier [3] summarize the difference between manual and automated testing by suggesting that automation should be used to prevent further errors in working modules, while manual testing is better suited for finding new and unexpected errors. However, how

and where the test automation should be used is not so straightforward issue, as the application of test automation seems to be a strongly diversified area of interest. The application of test automation has been studied for example in test case generation [27, 28], GUI testing [29, 30] and workflow simulators [31, 32] to name a few areas. Also according to Bertolino [33], test automation is a significant area of interest in current testing research, with a focus on improving the degree of automation by developing advanced techniques for generating the test inputs, or by finding support procedures such as error report generation to ease the supplemental workload. According to the same study, one of the dreams involving software testing is 100% automated testing. However, for example Bach's [23] study observes that this cannot be achieved, as all automation ultimately requires human intervention, if for nothing else, then at least to diagnose results and maintain automation cases.

The pressure to create resource savings are in many case the main argument for test automation. A simple and straightforward solution for building automation is to apply test automation just on the test cases and tasks that were previously done manually [8]. However, this approach is usually unfeasible. As Persson and Yilmaztürk [26] note, the establishment of automated testing is a costly, high risk project with several real possibilities for failure, commonly called as "pitfalls". One of the most common reasons why creating test automation fails, is that the software is not designed and implemented for testability and reusability, which leads to architecture and tools with low reusability and high maintenance costs. In reality, test automation sets several requisites on a project and has clear enablers and disablers for its suitability [4, 24]. In some reported cases [27, 34, 35], it was observed that the application of test automation with an ill-suited process model may be even harmful to the overall process in terms of productivity or cost-effectiveness.

Models for estimating testing automation costs, for example by Ramler and Wolfmaier [3], support decision-making in the tradeoff between automated and manual testing. Berner et al. [8] also estimate that most of the test cases in one project are run at least five times, and one fourth over 20 times. Therefore the test cases, which are done constantly like smoke tests, component tests and integration tests, seem like ideal place to build test automation. In any case, there seems to be a consensus that test automation is a plausible tool for enhancing quality, and consequently, reducing the software development costs in the long run if used correctly.

Our earlier research on the software test automation [4] has established that test automation is not as straightforward to implement as it may seem. There are several characteristics which enable or disable the applicability of test automation. In this study, our decision was to study a larger group of industry organizations and widen the perspective for further analysis. The objective is to observe, how the companies have implemented test automation and how they have responded to the issues and obstacles that affect its suitability in practice. Another objective is to analyze whether we can identify new kind of hindrances to the application of test automation and based on these findings, offer guidelines on what aspects should be taken into account when implementing test automation in practice.

3. Research Process

3.1. Research Population and Selection of the Sample. The population of the study consisted of organization units (OUs). The standard ISO/IEC 15504-1 [36] specifies an organizational unit (OU) as a part of an organization that is the subject of an assessment. An organizational unit deploys one or more processes that have a coherent process context and operates within a coherent set of business goals. An organizational unit is typically part of a larger organization, although in a small organization, the organizational unit may be the whole organization.

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 initial population and population criteria were decided based on the prior research on the subject. The sample for the first interview round consisted of 12 OUs, which were technically high level units, professionally producing software as their main process. This sample also formed the focus group of our study. Other selection criteria for the sample were based on the polar type selection [37] to cover different types of organizations, for example different businesses, different sizes of the company, and different kinds of operation. Our objective of using this approach was to gain a deep understanding of the cases and to identify, as broadly as possible, the factors and features that have an effect on software testing automation in practice.

For the second round and the survey, the sample was expanded by adding OUs to the study. Selecting the sample was demanding because comparability is not determined by the company or the organization but by comparable processes in the OUs. With the help of national and local authorities (the network of the Finnish Funding Agency for Technology and Innovation) we collected a population of 85 companies. Only one OU from each company was accepted to avoid the bias of over-weighting large companies. Each OU surveyed was selected from a company according to the population criteria. For this round, the sample size was expanded to 31 OUs, which also included the OUs from the first round. The selection for expansion was based on probability sampling; the additional OUs were randomly entered into our database, and every other one was selected for the survey. In the third round, the same sample as in the first round was interviewed. Table 1 introduces the business domains, company sizes and operating areas of our focus OUs. The company size classification is taken from [38].

3.2. Interview Rounds. The data collection consisted of three interview rounds. During the first interview round, the designers responsible for the overall software structure and/or module interfaces were interviewed. If the OU did not have separate designers, then the interviewed person was selected from the programmers based on their role in

Table 1: Description of the interviewed focus OUs (see also the appendix).

OU	Business	Company size[1]/Operation
Case A	MES[1] producer and electronics manufacturer	Small/National
Case B	Internet service developer and consultant	Small/National
Case C	Logistics software developer	Large/National
Case D	ICT consultant	Small/National
Case E	Safety and logistics system developer	Medium/National
Case F	Naval software system developer	Medium/International
Case G	Financial software developer	Large/National
Case H	MES[1] producer and logistics service systems provider	Medium/International
Case I	SME[2] business and agriculture ICT service provider	Small/National
Case J	Modeling software developer	Large/International
Case K	ICT developer and consultant	Large/International
Case L	Financial software developer	Large/International

[1] Manufacturing Execution System; [2] Small and Medium-sized Enterprise, definition [38].

the process to match as closely as possible to the desired responsibilities. The interviewees were also selected so that they came from the same project, or from positions where the interviewees were working on the same product. The interviewees were not specifically told not to discuss the interview questions together, but this behavior was not encouraged either. In a case where an interviewee asked for the questions or interview themes beforehand, the person was allowed access to them in order to prepare for the meeting. The interviews in all three rounds lasted about an hour and had approximately 20 questions related to the test processes or test organizations. In two interviews, there was also more than one person present.

The decision to interview designers in the first round was based on the decision to gain a better understanding on the test automation practice and to see whether our hypothesis based on our prior studies [4, 5, 12–14] and supplementing literature review were still valid. During the first interview round, we interviewed 12 focus OUs, which were selected to represent different polar types in the software industry. The interviews contained semi-structured questions and were tape-recorded for qualitative analysis. The initial analysis of the first round also provided ingredients for the further elaboration of important concepts for the latter rounds. The interview rounds and the roles of the interviewees in the case OUs are described in Table 2.

The purpose of the second combined interview and survey round was to collect multiple choice survey data and answers to open questions which were based on the first round interviews. These interviews were also tape-recorded for the qualitative analysis of the focus OUs, although the main data collection method for this round was a structured survey. In this round, project or testing managers from 31 OUs, including the focus OUs, were interviewed. The objective was to collect quantitative data on the software testing process, and further collect material on different testing topics, such as software testing and development. The collected survey data could also be later used to investigate observations made from the interviews and vice versa, as suggested in [38]. Managers were selected for this round, as they tend to have more experience on software projects, and have a better understanding of organizational and corporation level concepts and the overall software process beyond project-level activities.

The interviewees of the third round were testers or, if the OU did not have separate testers, programmers who were responsible for the higher-level testing tasks. The interviews in these rounds were also semi-structured and concerned the work of the interviewees, problems in testing (e.g., increasing complexity of the systems), the use of software components, the influence of the business orientation, testing resources, tools, test automation, outsourcing, and customer influence for the test processes.

The themes in the interview rounds remained similar, but the questions evolved from general concepts to more detailed ones. Before proceeding to the next interview round, all interviews with the focus OUs were transcribed and analyzed because new understanding and ideas emerged during the data analysis. This new understanding was reflected in the next interview rounds. The themes and questions for each of the interview rounds can be found on the project website http://www2.it.lut.fi/project/MASTO/.

3.3. Grounded Analysis Method. The grounded analysis was used to provide further insight into the software organizations, their software process and testing policies. By interviewing people of different positions from the production organization, the analysis could gain additional information on testing- and test automation-related concepts like different testing phases, test strategies, testing tools and case selection methods. Later this information could be compared between organizations, allowing hypotheses on test automation applicability and the test processes themselves.

The grounded theory method contains three data analysis steps: open coding, axial coding and selective coding. The objective for open coding is to extract the categories from the data, whereas axial coding identifies the connections between the categories. In the third phase, selective coding, the core category is identified and described [39]. In practice, these

Table 2: Interviewee roles and interview rounds.

Round type	Number of interviews	Interviewee role	Description
(1) Semistructured	12 focus OUs	Designer or Programmer	The interviewee is responsible for software design or has influence on how software is implemented
(2) Structured/ Semistructured	31 OUs quantitative, including 12 focus OUs qualitative	Project or testing manager	The interviewee is responsible for software development projects or test processes of software products
(3) Semistructured	12 focus OUs	Tester	The interviewee is a dedicated software tester or is responsible for testing the software product

steps overlap and merge because the theory development process proceeds iteratively. Additionally, Strauss and Corbin [40] state that 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.

The objective of open coding is to classify the data into categories and identify leads in the data, as shown in Table 3. The interview data is classified to categories based on the main issue, with observation or phenomenon related to it being the codified part. In general, the process of grouping concepts that seem to pertain to the same phenomena is called categorizing, and it is done to reduce the number of units to work with [40]. In our study, this was done using ATLAS.ti software [41]. The open coding process started with "seed categories" [42] that were formed from the research question and objectives, based on the literature study on software testing and our prior observations [4, 5, 12–14] on software and testing processes. Some seed categories, like "knowledge management", "service-orientation" or "approach for software development" were derived from our earlier studies, while categories like "strategy for testing", "outsourcing", "customer impact" or "software testing tools" were taken from known issues or literature review observations.

The study followed the approach introduced by Seaman [43], which 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 the open coding, we added new categories and merged existing categories to others if they seemed unfeasible or if we found a better generalization. Especially at the beginning of the analysis, the number of categories and codes quickly accumulated and the total number of codes after open coding amounted to 164 codes in 12 different categories. Besides the test process, software development in general and test automation, these categories also contained codified observations on such aspects as the business orientation, outsourcing, and product quality.

After collecting the individual observations to categories and codes, the categorized codes were linked together based on the relationships observed in the interviews. For example, the codes "Software process: Acquiring 3rd party modules", "Testing strategy: Testing 3rd party modules", and "Problem: Knowledge management with 3rd party modules" were clearly related and therefore we connected them together in axial coding. The objective of axial coding is to further develop categories, their properties and dimensions, and find causal, or any kinds of, connections between the categories and codes.

For some categories, the axial coding also makes it possible to define dimension for the phenomenon, for example "Personification-Codification" for "Knowledge management strategy", where every property could be defined as a point along the continuum defined by the two polar opposites. For the categories that are given dimension, the dimension represented the locations of the property or the attribute of a category [40]. Obviously for some categories, which were used to summarize different observations like enhancement proposals or process problems, defining dimensions was unfeasible. We considered using dimensions for some categories like "criticality of test automation in testing process" or "tool sophistication level for automation tools" in this study, but discarded them later as they yielded only little value to the study. This decision was based on the observation that values of both dimensions were outcomes of the applied test automation strategy, having no effect on the actual suitability or applicability of test automation to the organization's test process.

Our approach for analysis of the categories included Within-Case Analysis and Cross-Case-Analysis, as specified by Eisenhardt [37]. We used the tactic of selecting dimensions and properties with within-group similarities coupled with inter-group differences [37]. In this strategy, our team isolated one phenomenon that clearly divided the organizations to different groups, and searched for explaining differences and similarities from within these groups. Some of the applied features were, for example, the application of agile development methods, the application of test automation, the size [38] difference of originating companies and service orientation. As for one central result, the appropriateness of OU as a comparison unit was confirmed based on our size difference-related observations on the data; the within-group- and inter-group comparisons did yield results in which the company size or company policies did not have strong influence, whereas the local, within-unit policies did. In addition, the internal activities observed in OUs were similar regardless of the originating company size, meaning that in our study the OU comparison was indeed feasible approach.

We established and confirmed each chain of evidence in this interpretation method by discovering sufficient citations or finding conceptually similar OU activities from the case transcriptions. Finally, in the last phase of the analysis,

Table 3: Open coding of the interview data.

Interview transcript	Codes, Category: Code
"Well, I would hope for *stricter control or management for implementing our testing strategy*, as I am not sure if our *testing covers everything and is it sophisticated enough*. On the other hand, we do have *strictly limited resources, so it can be enhanced only to some degree*, we cannot test everything. And perhaps, recently we have had, in the newest versions, some regression testing, going through all features, seeing if nothing is broken, but *in several occasions this has been left unfinished because time has run out*. So there, on that issue we should focus."	Enhancement proposal: Developing testing strategy Strategy for testing: Ensuring case coverage Problem: Lack of resources Problem: Lack of time

in selective coding, our objective was to identify the core category [40]—a central phenomenon—and systematically relate it to other categories and generate the hypothesis and the theory. In this study, we consider test automation in practice as the core category, to which all other categories were related as explaining features of applicability or feasibility.

The general rule in grounded theory is to sample until theoretical saturation is reached. This means (1) no new or relevant data seem to emerge regarding a category, (2) the category development is dense, insofar as all of the paradigm elements are accounted for, along with variation and process, and (3) the relationships between categories are well established and validated [40]. In our study, the saturation was reached during the third round, where no new categories were created, merged, or removed from the coding. Similarly, the attribute values were also stable, that is, the already discovered phenomena began to repeat themselves in the collected data. As an additional way to ensure the validity of our study and avoid validity threats [44], four researchers took part in the data analysis. The bias caused by researchers was reduced by combining the different views of the researchers (observer triangulation) and a comparison with the phenomena observed in the quantitative data (methodological triangulation) [44, 45].

3.4. The Survey Instrument Development and Data Collection. The survey method described by Fink and Kosecoff [46] was used as the research method in the second round. An objective for a survey is to collect information from people about their feelings and beliefs. Surveys are most appropriate when information should come directly from the people [46]. Kitchenham et al. [47] divide comparable survey studies into exploratory studies from which only weak conclusions can be drawn, and confirmatory studies from which strong conclusions can be drawn. We consider this study as an exploratory, observational, and cross-sectional study that explores the phenomenon of software testing automation in practice and provides more understanding to both researchers and practitioners.

To obtain reliable measurements in the survey, a validated instrument was needed, but such an instrument was not available in the literature. However, Dybå [48] has developed an instrument for measuring the key factors of success in software process improvement. Our study was constructed based on the key factors of this instrument, and supplemented with components introduced in the standards ISO/IEC 29119 [11] and 25010 [49]. We had the possibility to use the current versions of the new standards because one of the authors is a member of the JTC1/SC7/WG26, which is developing the new software testing standard. Based on these experiences a measurement instrument derived from the ISO/IEC 29119 and 25010 standards was used.

The survey consisted of a questionnaire (available at http://www2.it.lut.fi/project/MASTO/) and a face-to-face interview. Selected open-ended questions were located at the end of the questionnaire to cover some aspects related to our qualitative study. The classification of the qualitative answers was planned in advance.

The questionnaire was planned to be answered during the interview to avoid missing answers because they make the data analysis complicated. All the interviews were tape-recorded, and for the focus organizations, further qualitatively analyzed with regard to the additional comments made during the interviews. Baruch [50] also states that the average response rate for self-assisted questionnaires is 55.6%, and when the survey involves top management or organizational representatives the response rate is 36.1%. In this case, a self-assisted, mailed questionnaire would have led to a small sample. For these reasons, it was rejected, and personal interviews were selected instead. The questionnaire was piloted with three OUs and four private persons.

If an OU had more than one respondent in the interview, they all filled the same questionnaire. Arranging the interviews, traveling and interviewing took two months of calendar time. Overall, we were able to accomplish 0.7 survey interviews per working day on an average. One researcher conducted 80% of the interviews, but because of the overlapping schedules also two other researchers participated in the interviews. Out of the contacted 42 OUs, 11 were rejected because they did not fit the population criteria in spite of the source information, or it was impossible to fit the interview into the interviewee's schedule. In a few individual cases, the reason for rejection was that the organization refused to give an interview. All in all, the response rate was, therefore, 74%.

4. Testing and Test Automation in Surveyed Organizations

4.1. General Information of the Organizational Units. The interviewed OUs were parts of large companies (55%) and small and medium-sized enterprises (45%). The OUs belonged to companies developing information systems (11 OUs), IT services (5 OUs), telecommunication (4 OUs),

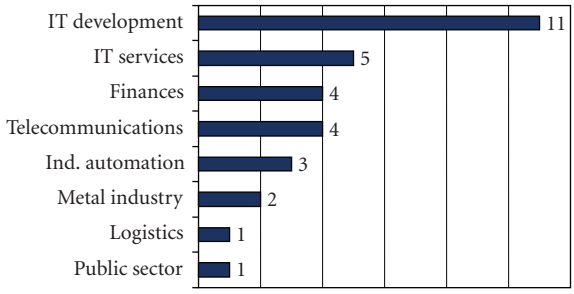

Figure 1: Application domains of the companies.

Table 4: Interviewed OUs.

	Max.	Min.	Median
Number of employees in the company.	350 000	4	315
Number of SW developers and testers in the OU.	600	0[1]	30
Percentage of automation in testing.	90	0	10
Percentage of agile (reactive, iterative) versus plan driven methods in projects.	100	0	30
Percentage of existing testers versus resources need.	100	10	75
How many percent of the development effort is spent on testing?	70	0[2]	25

[1] 0 means that all of the OUs developers and testers are acquired from 3rd parties.
[2] 0 means that no project time is allocated especially for testing.

finance (4 OUs), automation systems (3 OUs), the metal industry (2 OUs), the public sector (1 OU), and logistics (1 OU). The application domains of the companies are presented in Figure 1. Software products represented 63% of the turnover, and services (e.g., consulting, subcontracting, and integration) 37%.

The maximum number of personnel in the companies to which the OUs belonged was 350 000, the minimum was four, and the median was 315. The median of the software developers and testers in the OUs was 30 persons. OUs applied automated testing less than expected, the median of the automation in testing being 10%. Also, the interviewed OUs utilized agile methods less than expected: the median of the percentage of agile (reactive, iterative) versus plan driven methods in projects was 30%. The situation with human resources was better than what was expected, as the interviewees estimated that the amount of human resources in testing was 75%. When asking what percent of the development effort was spent on testing, the median of the answers was 25%. The cross-sectional situation of development and testing in the interviewed OUs is illustrated in Table 4.

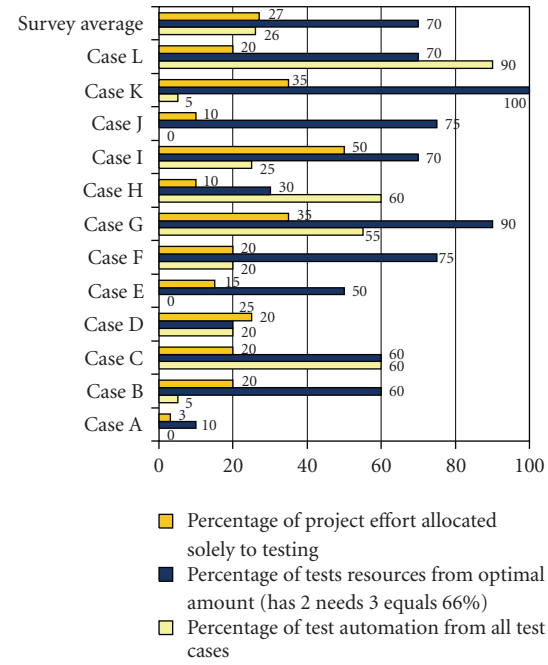

Figure 2: Amount of test resources and test automation in the focus organizations of the study and the survey average.

The amount of testing resources was measured by three figures; first the interviewee was asked to evaluate the percentage from total project effort allocated solely to testing. The survey average was 27%, the maximum being 70% and the minimum 0%, meaning that the organization relied solely on testing efforts carried out in parallel with development. The second figure was the amount of test resources compared to the organizational optimum. In this figure, if the company had two testers and required three, it would have translated as 66% of resources. Here the average was 70%; six organizations (19%) reported 100% resource availability. The third figure was the number of automated test cases compared to all of the test cases in all of the test phases the software goes through before its release. The average was 26%, varying between different types of organizations and project types. The results are presented in Figure 2, in which the qualitative study case OUs are also presented for comparison. The detailed descriptions for each case organization are available in the appendix.

4.2. General Testing Items. The survey interviewed 31 organization managers from different types of software industry. The contributions of the interviewees were measured using a five-point Likert scale where 1 denoted "I fully disagree" and 5 denoted "I fully agree". The interviewees emphasized that quality is built in development (4.3) rather than in testing (2.9). Then the interviewees were asked to estimate their organizational testing practices according to the new testing standard ISO/IEC 29119 [11], which identifies four main levels for testing processes: the test policy, test strategy, test management and testing. The test policy is the company level guideline which defines the management, framework

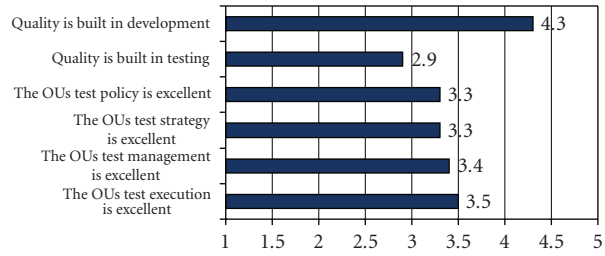

Figure 3: Levels of testing according to the ISO/IEC 29119 standard.

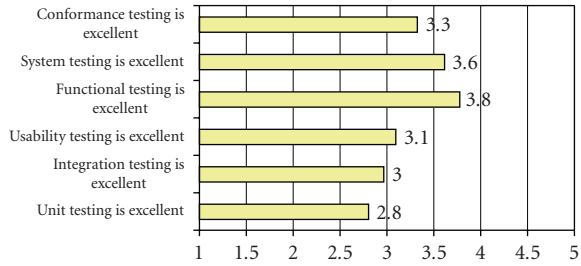

Figure 4: Testing phases in the software process.

and general guidelines, the test strategy is an adaptive model for the preferred test process, test management is the control level for testing in a software project, and finally, testing is the process of conducting test cases. The results did not make a real difference between the lower levels of testing (test management level and test levels) and higher levels of testing (organizational test policy and organizational test strategy). All in all, the interviewees were rather satisfied with the current organization of testing. The resulted average levels from quantitative survey are presented in Figure 3.

Besides the organization, the test processes and test phases were also surveyed. The five-point Likert scale with the same one to five—one being fully disagree and five fully agree—grading method was used to determine the correctness of different testing phases. Overall, the latter test phases—system, functional testing—were considered excellent or very good, whereas the low level test phases such as unit testing and integration received several low-end scores. The organizations were satisfied or indifferent towards all test phases, meaning that there were no strong focus areas for test organization development. However, based on these results it seems plausible that one effective way to enhance testing would be to support low-level testing in unit and integration test phases. The results are depicted in Figure 4.

Finally, the organizations surveyed were asked to rate their testing outcomes and objectives (Figure 5). The first three items discussed the test processes of a typical software project. There seems to be a strong variance in testing schedules and time allocation in the organizations. The outcomes 3.2 for schedule and 3.0 for time allocation do not give any information by themselves, and overall, the direction of answers varied greatly between "Fully disagree" and "Fully agree". However, the situation with test processes

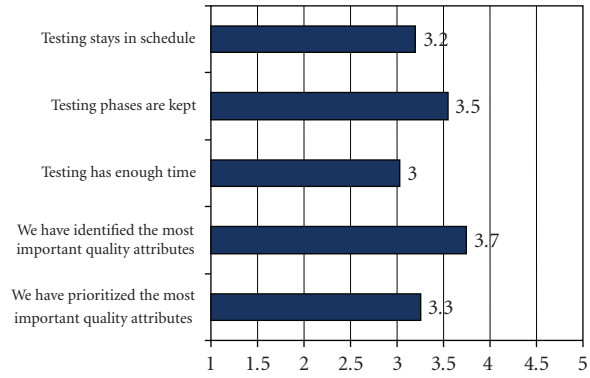

Figure 5: Testing process outcomes.

was somewhat better; the result 3.5 may also not be a strong indicator by itself, but the answers had only little variance, 20 OUs answering "somewhat agree" or "neutral". This indicates that even if the time is limited and the project schedule restricts testing, the testing generally goes through the normal, defined, procedures.

The fourth and fifth items were related to quality aspects, and gave insights into the clarity of testing objectives. The results of 3.7 for the identification of quality attributes indicate that organizations tend to have objectives for the test processes and apply quality criteria in development. However, the prioritization of their quality attributes is not as strong (3.3) as identification.

4.3. Testing Environment. The quality aspects were also reflected in the employment of systematic methods for the testing work. The majority (61%) of the OUs followed a systematic method or process in the software testing, 13% followed one partially, and 26% of the OUs did not apply any systematic method or process in testing. Process practices were derived from, for example, TPI (Test Process Improvement) [51] or the Rational Unified Process (RUP) [52]. Few Agile development process methods such as Scrum [53] or XP (eXtreme Programming) [54] were also mentioned.

A systematic method is used to steer the software project, but from the viewpoint of testing, the process also needs an infrastructure on which to operate. Therefore, the OUs were asked to report which kind of testing tools they apply to their typical software processes. The test management tools, tools which are used to control and manage test cases and allocate testing resources to cases, turned out to be the most popular category of tools; 15 OUs out of 31 reported the use of this type of tool. The second in popularity were manual unit testing tools (12 OUs), which were used to execute test cases and collect test results. Following them were tools to implement test automation, which were in use in 9 OUs, performance testing tools used in 8 OUs, bug reporting tools in 7 OUs and test design tools in 7 OUs. Test design tools were used to create and design new test cases. The group of other tools consisted of, for example, electronic measurement devices, test report generators, code analyzers, and project

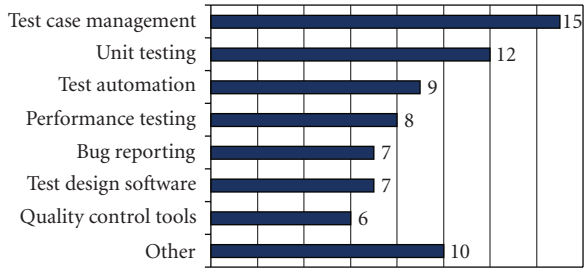

Figure 6: Popularity of the testing tools according to the survey.

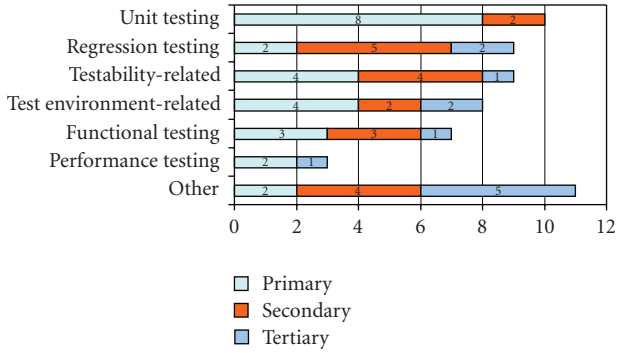

Figure 7: The three most efficient application areas of test automation tools according to the interviewees.

management tools. The popularity of the testing tools in different survey organizations is illustrated in Figure 6.

The respondents were also asked to name and explain the three most efficient application areas of test automation tools. Both the format of the open-ended questions and the classification of the answers were based on the like best (LB) technique adopted from Fink and Kosecoff [46]. According to the LB technique, respondents were asked to list points they considered the most efficient. The primary selection was the area in which the test automation would be the most beneficial to the test organization, the secondary one is the second best area of application, and the third one is the third best area. The interviewees were also allowed to name only one or two areas if they were unable to decide on three application areas. The results revealed the relative importance of software testing tools and methods.

The results are presented in Figure 7. The answers were distributed rather evenly between different categories of tools or methods. The most popular category was unit testing tools or methods (10 interviewees). Next in line were regression testing (9), tools to support testability (9), test environment tools and methods (8), and functional testing (7). The group "others" (11) consisted of conformance testing tools, TTCN-3 (*Testing and Test Control Notation version 3*) tools, general test management tools such as document generators and methods of unit and integration testing. The most popular category, unit testing tools or methods, also received the most primary application area nominations. The most common secondary area of application was regression testing. Several categories ranked third, but concepts such as regression testing, and test environment-related aspects such as document generators were mentioned more than once. Also testability-related concepts—module interface, conformance testing—or functional testing—verification, validation tests—were considered feasible implementation areas for test automation.

4.4. Summary of the Survey Findings. The survey suggests that interviewees were rather satisfied with their test policy, test strategy, test management, and testing, and did not have any immediate requirements for revising certain test phases, although low-level testing was slightly favoured in the development needs. All in all, 61% of the software companies followed some form of a systematic process or method in testing, with an additional 13% using some established procedures or measurements to follow the process efficiency. The systematic process was also reflected in the general approach to testing; even if the time was limited, the test process followed a certain path, applying the test phases regardless of the project limitations.

The main source of the software quality was considered to be in the development process. In the survey, the test organizations used test automation on an average on 26% of their test cases, which was considerably less than could be expected based on the literature. However, test automation tools were the third most common category of test-related tools, commonly intended to implement unit and regression testing. As for the test automation itself, the interviewees ranked unit testing tools as the most efficient tools of test automation, regression testing being the most common secondary area of application.

5. Test Automation Interviews and Qualitative Study

Besides the survey, the test automation concepts and applications were analyzed based on the interviews with the focus organizations. The grounded theory approach was applied to establish an understanding of the test automation concepts and areas of application for test automation in industrial software engineering. The qualitative approach was applied in three rounds, in which a developer, test manager and tester from 12 different case OUs were interviewed. Descriptions of the case OUs can be found in the appendix.

In theory-creating inductive research [55], the central idea is that researchers constantly compare theory and data iterating with a theory which closely fits the data. Based on the grounded theory codification, the categories identified were selected in the analysis based on their ability to differentiate the case organizations and their potential to explain the differences regarding the application of test automation in different contexts. We selected the categories so as to explore the types of automation applications and the compatibility of test automation services with the OUs testing organization. We conceptualized the most common test automation concepts based on the coding and further elaborated them to categories to either cater the essential features such as their role in the overall software process or

their relation to test automation. We also concentrated on the OU differences in essential concepts such as automation tools, implementation issues or development strategies. This conceptualization resulted to the categories listed in Table 5.

The category "Automation application" describes the areas of software development, where test automation was applied successfully. This category describes the testing activities or phases which apply test automation processes. In the case where the test organization did not apply automation, or had so far only tested it for future applications, this category was left empty. The application areas were generally geared towards regression and stress testing, with few applications of functionality and smoke tests in use.

The category "Role in software process" is related to the objective for which test automation was applied in software development. The role in the software process describes the objective for the existence of the test automation infrastructure; it could, for example, be in quality control, where automation is used to secure module interfaces, or in quality assurance, where the operation of product functionalities is verified. The usual role for the test automation tools was in quality control and assurance, the level of application varying from third party-produced modules to primary quality assurance operations. On two occasions, the role of test automation was considered harmful to the overall testing outcomes, and on one occasion, the test automation was considered trivial, with no real return on investments compared to traditional manual testing.

The category "Test automation strategy" is the approach to how automated testing is applied in the typical software processes, that is, the way the automation was used as a part of the testing work, and how the test cases and overall test automation strategy were applied in the organization. The level of commitment to applying automation was the main dimension of this category, the lowest level being individual users with sporadic application in the software projects, and the highest being the application of automation to the normal, everyday testing infrastructure, where test automation was used seamlessly with other testing methods and had specifically assigned test cases and organizational support.

The category of "Automation development" is the general category for OU test automation development. This category summarizes the ongoing or recent efforts and resource allocations to the automation infrastructure. The type of new development, introduction strategies and current development towards test automation are summarized in this category. The most frequently chosen code was "general increase of application", where the organization had committed itself to test automation, but had no clear idea of how to develop the automation infrastructure. However, one OU had a development plan for creating a GUI testing environment, while two organizations had just recently scaled down the amount of automation as a result of a pilot project. Two organizations had only recently introduced test automation to their testing infrastructure.

The category of "Automation tools" describes the types of test automation tools that are in everyday use in the OU. These tools are divided based on their technological finesse, varying from self-created drivers and stubs to individual proof-of-concept tools with one specified task to test suites where several integrated components are used together for an effective test automation environment. If the organization had created the tools by themselves, or customized the acquired tools to the point of having new features and functionalities, the category was supplemented with a notification regarding in-house-development.

Finally, the category of "Automation issues" includes the main hindrances which are faced in test automation within the organization. Usually, the given issue was related to either the costs of test automation or the complexity of introducing automation to the software projects which have been initially developed without regards to support for automation. Some organizations also considered the efficiency of test automation to be the main issue, mostly contributing to the fact that two of them had just recently scaled down their automation infrastructure. A complete list of test automation categories and case organizations is given in Table 6.

We elaborated further these properties we observed from the case organizations to create hypotheses for the test automation applicability and availability. These resulting hypotheses were shaped according to advice given by Eisenhardt [37] for qualitative case studies. For example, we perceived the quality aspect as really important for the role of automation in software process. Similarly, the resource needs, especially costs, were much emphasized in the automation issues category. The purpose of the hypotheses below is to summarize and explain the features of test automation that resulted from the comparison of differences and similarities between the organizations.

Hypothesis 1 (Test automation should be considered more as a quality control tool rather than a frontline testing method). The most common area of application observed was functionality verification, that is, regression testing and GUI event testing. As automation is time-consuming and expensive to create, these were the obvious ways to create test cases which had the minimal number of changes per development cycle. By applying this strategy, organizations could set test automation to confirm functional properties with suitable test cases, and acquire such benefits as support for change management and avoid unforeseen compatibility issues with module interfaces.

> "*Yes, regression testing, especially automated. It is not manually "hammered in" every time, but used so that the test sets are run, and if there is anything abnormal, it is then investigated.*"—Manager, Case G

> "*... had we not used it [automation tests], it would have been suicidal.*"—Designer, Case D

> "*It's [automated stress tests] good for showing bad code, how efficient it is and how well designed ... stress it enough and we can see if it slows down or even breaks completely.*"—Tester, Case E

Table 5: Test automation categories.

Category	Definition
Automation application	Areas of application for test automation in the software process
Role in software process	The observed roles of test automation in the company software process and the effect of this role
Test automation strategy	The observed method for selecting the test cases where automation is applied and the level of commitment to the application of test automation in the organizations
Automation development	The areas of active development in which the OU is introducing test automation
Automation tools	The general types of test automation tools applied
Automation issues	The items that hinder test automation development in the OU

However, there seemed to be some contradicting considerations regarding the applicability of test automation. Cases F, J, and K had recently either scaled down their test automation architecture or considered it too expensive or inefficient when compared to manual testing. In some cases, automation was also considered too bothersome to configure for a short-term project, as the system would have required constant upkeep, which was an unnecessary addition to the project workload.

> "*We really have not been able to identify any major advancements from it [test automation].*"—Tester, Case J

> "*It [test automation] just kept interfering.*"—Designer, Case K

Both these viewpoints indicated that test automation should not be considered a "frontline" test environment for finding errors, but rather a quality control tool to maintain functionalities. For unique cases or small projects, test automation is too expensive to develop and maintain, and it generally does not support single test cases or explorative testing. However, it seems to be practical in larger projects, where verifying module compatibility or offering legacy support is a major issue.

Hypothesis 2 (Maintenance and development costs are common test automation hindrances that universally affect all test organizations regardless of their business domain or company size). Even though the case organizations were selected to represent different types of organizations, the common theme was that the main obstacles in automation adoption were development expenses and upkeep costs. It seemed to make no difference whether the organization unit belonged to a small or large company, as in the OU levels they shared common obstacles. Even despite the maintenance and development hindrances, automation was considered a feasible tool in many organizations. For example, Cases I and L pursued the development of some kind of automation to enhance the testing process. Similarly, Cases E and H, which already had a significant number of test automation cases, were actively pursuing a larger role for automated testing.

> "*Well, it [automation] creates a sense of security and controllability, and one thing that is easily underestimated is its effect on performance and optimization. It requires regression tests to confirm that if something is changed, the whole thing does not break down afterwards.*"—Designer, Case H

In many cases, the major obstacle for adopting test automation was, in fact, the high requirements for process development resources.

> "*Shortage of time, resources … we have the technical ability to use test automation, but we don't.*" —Tester, Case J

> "*Creating and adopting it, all that it takes to make usable automation … I believe that we don't put any effort into it because it will end up being really expensive.*" —Designer, Case J

In Case J particularly, the OU saw no incentive in developing test automation as it was considered to offer only little value over manual testing, even if they otherwise had no immediate obstacles other than implementation costs. Also cases F and K reported similar opinions, as they both had scaled down the amount of automation after the initial pilot projects.

> "*It was a huge effort to manually confirm why the results were different, so we took it [automation] down.*"—Tester, Case F

> "*Well, we had gotten automation tools from our partner, but they were so slow we decided to go on with manual testing.*"—Tester, Case K

Hypothesis 3 (Test automation is applicable to most of the software processes, but requires considerable effort from the organization unit). The case organizations were selected to represent the polar types of software production operating in different business domains. Out of the focus OUs, there were four software development OUs, five IT service OUs, two OUs from the finance sector and one logistics OU. Of these OUs, only two did not have any test automation, and two others had decided to strategically abandon their test automation infrastructure. Still, the business domains for the remaining organizations which applied test automation were

Table 6: Test automation categories affecting the software process in case OUs.

OU	Category					
	Automation application	Role in software process	Test automation strategy	Automation development	Automation tools	Automation issues
Case A	GUI testing, regression testing	Functionality verification	Part of the normal test infrastructure	General increase of application	Individual tools, test suite, in-house development	Complexity of adapting automation to test processes
Case B	Performance, smoke testing	Quality control tool	Part of the normal test infrastructure	GUI testing, unit testing	Individual tools, in-house development	Costs of automation implementation
Case C	Functionality, regression testing, documentation automation	Quality control tool	Part of the normal test infrastructure	General increase of application	Test suite, in-house development	Cost of automation maintenance
Case D	Functionality testing	Quality control for secondary modules	Project-related cases	Upkeep for existing parts	Individual tools	Costs of automation implementation
Case E	System stress testing	Quality assurance tool	Part of the normal test infrastructure	General increase of application	Test suite	Costs of implementing new automation
Case F	Unit and module testing, documentation automation	QC, overall effect harmful	Individual users	Recently scaled down	Individual tools	Manual testing seen more efficient
Case G	Regression testing for use cases	Quality assurance tool	Part of the normal test infrastructure	General increase of application	Test suite	Cost of automation maintenance
Case H	Regression testing for module interfaces	Quality control for secondary modules	Part of the normal test infrastructure	General increase of application	Test suite, in-house development	Underestimation of the effect of automated testing on quality
Case I	Functionality testing	Quality control tool	Project-related cases	Application pilot in development	Proof-of-concept tools	Costs of automation implementation
Case J	Automation not in use	QA, no effect observed	Individual users	Application pilot in development	Proof-of-concept tools	No development incentive
Case K	Small scale system testing	QC, overall effect harmful	Individual users	Recently scaled down	Self-created tools; drivers and stubs	Manual testing seen more efficient
Case L	System stress testing	Verifies module compatibility	Project-related cases	Adapting automation to the testing strategy	Individual tools, in-house development	Complexity of adapting automation to test processes

heterogeneously divided, meaning that the business domain is not a strong indicator of whether or not test automation should be applied.

It seems that test automation is applicable as a test tool in any software process, but the amount of resources required for useful automation compared to the overall development resources is what determines whether or not automation should be used. As automation is oriented towards quality control aspects, it may be unfeasible to implement in small development projects where quality control is manageable with manual confirmation. This is plausible, as the amount of required resources does not seem to vary based on aspects beyond the OU characteristics, such as available company resources or testing policies applied. The feasibility of test automation seems to be rather connected to the actual software process objectives, and fundamentally to the decision whether the quality control aspects gained from test automation supersede the manual effort required for similar results.

> *"... before anything is automated, we should calculate the maintenance effort and estimate*

> *whether we will really save time, instead of just automating for automation's sake.*"—Tester, Case G

> "*It always takes a huge amount of resources to implement.*"—Designer, Case A

> "*Yes, developing that kind of test automation system is almost as huge an effort as building the actual project.*" —Designer, Case I

Hypothesis 4 (The available repertoire of testing automation tools is limited, forcing OUs to develop the tools themselves, which subsequently contributes to the application and maintenance costs). There were only a few case OUs that mentioned any commercial or publicly available test automation programs or suites. The most common approach to test automation tools was to first acquire some sort of tool for proof-of-concept piloting, then develop similar tools as in-house-production or extend the functionalities beyond the original tool with the OU's own resources. These resources for in-house-development and upkeep for self-made products are one of the components that contribute to the costs of applying and maintaining test automation.

> "*Yes, yes. That sort of [automation] tools have been used, and then there's a lot of work that we do ourselves. For example, this stress test tool . . .*"—Designer, Case E

> "*We have this 3rd party library for the automation. Well, actually, we have created our own architecture on top of it . . .*"—Designer, Case H

> "*Well, in [company name], we've-, we developed our own framework to, to try and get around some of these, picking which tests, which group of tests should be automated.*"—Designer, Case C

However, it should be noted that even if the automation tools were well-suited for the automation tasks, the maintenance still required significant resources if the software product to which it was connected was developing rapidly.

> "*Well, there is the problem [with automation tool] that sometimes the upkeep takes an incredibly large amount of time.*" —Tester, Case G

> "*Our system keeps constantly evolving, so you'd have to be constantly recording [maintaining tools]. . .*" —Tester, Case K

6. Discussion

An exploratory survey combined with interviews was used as the research method. The objective of this study was to shed light on the status of test automation and to identify improvement needs in and the practice of test automation. The survey revealed that the total effort spent on testing (median 25%) was less than expected. The median percentage (25%) of testing is smaller than the 50%–60% that is often mentioned in the literature [38, 39]. The comparable low percentage may indicate that that the resources needed for software testing are still underestimated even though testing efficiency has grown. The survey also indicated that companies used fewer resources on test automation than expected: on an average 26% of all of the test cases apply automation. However, there seems to be ambiguity as to which activities organizations consider test automation, and how automation should be applied in the test organizations. In the survey, several organizations reported that they have an extensive test automation infrastructure, but this did not reflect on the practical level, as in the interviews with testers particularly, the figures were considerably different. This indicates that the test automation does not have strong strategy in the organization, and has yet to reach maturity in several test organizations. Such concepts as quality assurance testing and stress testing seem to be particularly unambiguous application areas, as Cases E and L demonstrated. In Case E, the management did not consider stress testing an automation application, whereas testers did. Moreover, in Case L the large automation infrastructure did not reflect on the individual project level, meaning that the automation strategy may strongly vary between different projects and products even within one organization unit.

The qualitative study which was based on interviews indicated that some organizations, in fact, actively avoid using test automation, as it is considered to be expensive and to offer only little value for the investment. However, test automation seems to be generally applicable to the software process, but for small projects the investment is obviously oversized. One additional aspect that increases the investment are tools, which unlike in other areas of software testing, tend to be developed in-house or are heavily modified to suit specific automation needs. This development went beyond the localization process which every new software tool requires, extending even to the development of new features and operating frameworks. In this context it also seems plausible that test automation can be created for several different test activities. Regression testing, GUI testing or unit testing, activities which in some form exist in most development projects, all make it possible to create successful automation by creating suitable tools for the task, as in each phase can be found elements that have sufficient stability or unchangeability. Therefore it seems that the decision on applying automation is not only connected to the enablers and disablers of test automation [4], but rather on tradeoff of required effort and acquired benefits; In small projects or with low amount of reuse the effort becomes too much for such investment as applying automation to be feasible.

The investment size and requirements of the effort can also be observed on two other occasions. First, test automation should not be considered as an active testing tool for finding errors, but as a tool to guarantee the functionality of already existing systems. This observation is in line with those of Ramler and Wolfmaier [3], who discuss the necessity of a large number of repetitive tasks for the automation to supersede manual testing in cost-effectiveness, and of

Berner et al. [8], who notify that the automation requires a sound application plan and well-documented, simulatable and testable objects. For both of these requirements, quality control at module interfaces and quality assurance on system operability are ideal, and as it seems, they are the most commonly used application areas for test automation. In fact, Kaner [56] states that 60%–80% of the errors found with test automation are found in the development phase for the test cases, further supporting the quality control aspect over error discovery.

Other phenomena that increase the investment are the limited availability and applicability of automation tools. On several occasions, the development of the automation tools was an additional task for the automation-building organization that required the organization to allocate their limited resources to the test automation tool implementation. From this viewpoint it is easy to understand why some case organizations thought that manual testing is sufficient and even more efficient when measured in resource allocation per test case. Another approach which could explain the observed resistance to applying or using test automation was also discussed in detail by Berner et al. [8], who stated that organizations tend to have inappropriate strategies and overly ambitious objectives for test automation development, leading to results that do not live up to their expectations, causing the introduction of automation to fail. Based on the observations regarding the development plans beyond piloting, it can also be argued that the lack of objectives and strategy also affect the successful introduction processes. Similar observations of "automation pitfalls" were also discussed by Persson and Yilmaztürk [26] and Mosley and Posey [57].

Overall, it seems that the main disadvantages of testing automation are the costs, which include implementation costs, maintenance costs, and training costs. Implementation costs included direct investment costs, time, and human resources. The correlation between these test automation costs and the effectiveness of the infrastructure are discussed by Fewster [24]. If the maintenance of testing 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, making automation a bad investment for the organization. We observed this phenomenon in two case organizations. There is also a connection between implementation costs and maintenance costs [24]. If the testing automation system is designed with the minimization of maintenance costs in mind, the implementation costs increase, and vice versa. We noticed the phenomenon of costs preventing test automation development in six cases. The implementation of test automation seems to be possible to accomplish with two different approaches: by promoting either maintainability or easy implementation. If the selected focus is on maintainability, test automation is expensive, but if the approach promotes easy implementation, the process of adopting testing automation has a larger possibility for failure. This may well be due to the higher expectations and assumption that the automation could yield results faster when promoting implementation over maintainability, often leading to one of the automation pitfalls [26] or at least a low percentage of reusable automation components with high maintenance costs.

7. Conclusions

The objective of this study was to observe and identify factors that affect the state of testing, with automation as the central aspect, in different types of organizations. Our study included a survey in 31 organizations and a qualitative study in 12 focus organizations. We interviewed employees from different organizational positions in each of the cases.

This study included follow-up research on prior observations [4, 5, 12–14] on testing process difficulties and enhancement proposals, and on our observations on industrial test automation [4]. In this study we further elaborated on the test automation phenomena with a larger sample of polar type OUs, and more focused approach on acquiring knowledge on test process-related subjects. The survey revealed that test organizations use test automation only in 26% of their test cases, which was considerably less than could be expected based on the literature. However, test automation tools were the third most common category of test-related tools, commonly intended to implement unit and regression testing. The results indicate that adopting test automation in software organization is a demanding effort. The lack of existing software repertoire, unclear objectives for overall development and demands for resource allocation both for design and upkeep create a large threshold to overcome.

Test automation was most commonly used for quality control and quality assurance. In fact, test automation was observed to be best suited to such tasks, where the purpose was to secure working features, such as check module interfaces for backwards compatibility. However, the high implementation and maintenance requirements were considered the most important issues hindering test automation development, limiting the application of test automation in most OUs. Furthermore, the limited availability of test automation tools and the level of commitment required to develop a suitable automation infrastructure caused additional expenses. Due to the high maintenance requirements and low return on investments in small-scale application, some organizations had actually discarded their automation systems or decided not to implement test automation. The lack of a common strategy for applying automation was also evident in many interviewed OUs. Automation applications varied even within the organization, as was observable in the differences when comparing results from different stakeholders. In addition, the development strategies were vague and lacked actual objectives. These observations can also indicate communication gaps [58] between stakeholders of the overall testing strategy, especially between developers and testers.

The data also suggested that the OUs that had successfully implemented test automation infrastructure to cover the entire organization seemed to have difficulties in creating a continuance plan for their test automation development. After the adoption phases were over, there was an ambiguity about how to continue, even if the organization had decided

to further develop their test automation infrastructure. The overall objectives were usually clear and obvious—cost savings and better test coverage—but in practise there were only few actual development ideas and novel concepts. In the case organizations this was observed in the vagueness of the development plans: only one of the five OUs which used automation as a part of their normal test processes had development plans beyond the general will to increase the application.

The survey established that 61% of the software companies followed some form of a systematic process or method in testing, with an additional 13% using some established procedures or measurements to follow the process efficiency. The main source of software quality was considered to reside in the development process, with testing having much smaller impact in the product outcome. In retrospect of the test levels introduced in the ISO/IEC29119 standard, there seems to be no one particular level of the testing which should be the research and development interest for best result enhancements. However, the results from the self-assessment of the test phases indicate that low-level testing could have more potential for testing process development.

Based on these notions, the research and development should focus on uniform test process enhancements, such as applying a new testing approach and creating an organization-wide strategy for test automation. Another focus area should be the development of better tools to support test organizations and test processes in the low-level test phases such as unit or integration testing. As for automation, one tool project could be the development of a customizable test environment with a common core and with an objective to introduce less resource-intensive, transferable and customizable test cases for regression and module testing.

Appendix

Case Descriptions

Case A (Manufacturing execution system (MES) producer and electronics manufacturer). Case A produces software as a service (SaaS) for their product. The company is a small-sized, nationally operating company that has mainly industrial customers. Their software process is a plan-driven cyclic process, where the testing is embedded to the development itself, having only little amount of dedicated resources. This organization unit applied test automation as a user interface and regression testing tool, using it for product quality control. Test automation was seen as a part of the normal test strategy, universally used in all software projects. The development plan for automation was to generally increase the application, although the complexity of the software- and module architecture was considered major obstacle on the automation process.

Case B (Internet service developer and consultant). Case B organization offers two types of services; development of Internet service portals for the customers like communities and public sector, and consultation in the Internet service business domain. The origination company is small and operates on a national level. Their main resource on the test automation is in the performance testing as a quality control tool, although addition of GUI test automation has also been proposed. The automated tests are part of the normal test process, and the overall development plan was to increase the automation levels especially to the GUI test cases. However, this development has been hindered by the cost of designing and developing test automation architecture.

Case C (Logistics software developer). Case C organization focuses on creating software and services for their origin company and its customers. This organization unit is a part of a large-sized, nationally operating company with large, highly distributed network and several clients. The test automation is widely used in several testing phases like functionality testing, regression testing and document generation automation. These investments are used for quality control to ensure the software usability and correctness. Although the OU is still aiming for larger test automation infrastructure, the large amount of related systems and constant changes within the inter-module communications is causing difficulties in development and maintenance of the new automation cases.

Case D (ICT consultant). Case D organization is a small, regional software consultant company, whose customers mainly compose of small business companies and the public sector. Their organization does some software development projects, in which the company develops services and ICT products for their customers. The test automation comes mainly trough this channel, as the test automation is mainly used as a conformation test tool for the third party modules. This also restricts the amount of test automation to the projects, in which these modules are used. The company currently does not have development plans for the test automation as it is considered unfeasible investment for the OU this size, but they do invest on the upkeep of the existing tools as they have usage as a quality control tool for the acquired outsider modules.

Case E (Safety and logistics system developer). Case E organization is a software system developer for safety and logistics systems. Their products have high amount of safety critical features and have several interfaces on which to communicate with. The test automation is used as a major quality assurance component, as the service stress tests are automated to a large degree. Therefore the test automation is also a central part of the testing strategy, and each project has defined set of automation cases. The organization is aiming to increase the amount of test automation and simultaneously develop new test cases and automation applications for the testing process. The main obstacle for this development has so far been the costs of creating new automation tools and extending the existing automation application areas.

Case F (Naval software system developer). The Case F organization unit is responsible for developing and testing

naval service software systems. Their product is based on a common core, and has considerable requirements for compatibility with the legacy systems. This OU has tried test automation on several cases with application areas such as unit- and module testing, but has recently scaled down test automation for only support aspects such as the documentation automation. This decision was based on the resource requirements for developing and especially maintaining the automation system, and because the manual testing was in this context considered much more efficient as there were too much ambiguity in the automation-based test results.

Case G (Financial software developer). Case G is a part of a large financial organization, which operates nationally but has several internationally connected services due to their business domain. Their software projects are always aimed as a service portal for their own products, and have to pass considerable verification and validation tests before being introduced to the public. Because of this, the case organization has sizable test department when compared to other case companies in this study, and follows rigorous test process plan in all of their projects. The test automation is used in the regression tests as a quality assurance tool for user interfaces and interface events, and therefore embedded to the testing strategy as a normal testing environment. The development plans for the test automation is aimed to generally increase the amount of test cases, but even the existing test automation infrastructure is considered expensive to upkeep and maintain.

Case H (Manufacturing execution system (MES) producer and logistics service system provider). Case H organization is a medium-sized company, whose software development is a component for the company product. Case organization products are used in logistics service systems, usually working as a part of automated processes. The case organization applies automated testing as a module interface testing tool, applying it as a quality control tool in the test strategy. The test automation infrastructure relies on the in-house-developed testing suite, which enables organization to use the test automation to run daily tests to validate module conformance. Their approach on the test automation has been seen as a positive enabler, and the general trend is towards increasing automation cases. The main test automation disability is considered to be that the quality control aspect is not visible when working correctly and therefore the effect of test automation may be underestimated in the wider organization.

Case I (Small and medium-sized enterprise (SME) business and agriculture ICT-service provider). The case I organization is a small, nationally operating software company which operates on multiple business domain. Their customer base is heterogeneous, varying from finances to the agriculture and government services. The company is currently not utilizing test automation in their test process, but they have development plans for designing quality control automation. For this development they have had some individual proof-of-concept tools, but currently the overall testing resources limit the application process.

Case J (Modeling software developer). Case J organization develops software products for civil engineering and architectural design. Their software process is largely plan-driven with rigorous verification and validation processes in the latter parts of an individual project. Even though the case organization itself has not implemented test automation, on the corporate level there are some pilot projects where regression tests have been automated. These proof-of-concept-tools have been introduced to the case OU and there are intentions to apply them in the future, but there has so far been no incentive for adoption of the automation tools, delaying the application process.

Case K (ICT developer and consultant). Case K organization is a large, international software company which offers software products for several business domains and government services. Case organization has previously piloted test automation, but decided against adopting the system as it was considered too expensive and resource-intensive to maintain when compared to the manual testing. However, some of these tools still exist, used by individual developers along with test drivers and interface studs in unit- and regression testing.

Case L (Financial software developer). Case L organization is a large software provider for their corporate customer which operates on the finance sector. Their current approach on software process is plan-driven, although some automation features has been tested on a few secondary processes. The case organization does not apply test automation as is, although some module stress test cases have been automated as pilot tests. The development plan for test automation is to generally implement test automation as a part of their testing strategy, although amount of variability and interaction in the module interfaces is considered difficult to implement in test automation cases.

Acknowledgment

This study is a part of the ESPA project (http://www.soberit.hut.fi/espa/), funded by the Finnish Funding Agency for Technology and Innovation (project number 40125/08) and by the participating companies listed on the project web site.

References

[1] E. Kit, *Software Testing in the Real World: Improving the Process*, Addison-Wesley, Reading, Mass, USA, 1995.

[2] G. Tassey, "The economic impacts of inadequate infrastructure for software testing," RTI Project 7007.011, U.S. National Institute of Standards and Technology, Gaithersburg, Md, USA, 2002.

[3] R. Ramler and K. Wolfmaier, "Observations and lessons learned from automated testing," in *Proceedings of the International Workshop on Automation of Software Testing (AST '06)*, pp. 85–91, Shanghai, China, May 2006.

[4] K. Karhu, T. Repo, O. Taipale, and K. Smolander, "Empirical observations on software testing automation," in *Proceedings of the 2nd International Conference on Software Testing, Verification, and Validation (ICST '09)*, pp. 201–209, Denver, Colo, USA, April 2009.

[5] O. Taipale and K. Smolander, "Improving software testing by observing causes, effects, and associations from practice," in *Proceedings of the International Symposium on Empirical Software Engineering (ISESE '06)*, Rio de Janeiro, Brazil, September 2006.

[6] B. Shea, "Sofware testing gets new respect," *InformationWeek*, July 2000.

[7] E. Dustin, J. Rashka, and J. Paul, *Automated Software Testing: Introduction, Management, and Performance*, Addison-Wesley, Boston, Mass, USA, 1999.

[8] S. Berner, R. Weber, and R. K. Keller, "Observations and lessons learned from automated testing," in *Proceedings of the 27th International Conference on Software Engineering (ICSE '05)*, pp. 571–579, St. Louis, Mo, USA, May 2005.

[9] J. A. Whittaker, "What is software testing? And why is it so hard?" *IEEE Software*, vol. 17, no. 1, pp. 70–79, 2000.

[10] L. J. Osterweil, "Software processes are software too, revisited: an invited talk on the most influential paper of ICSE 9," in *Proceedings of the 19th IEEE International Conference on Software Engineering*, pp. 540–548, Boston, Mass, USA, May 1997.

[11] ISO/IEC and ISO/IEC 29119-2, "Software Testing Standard—Activity Descriptions for Test Process Diagram," 2008.

[12] O. Taipale, K. Smolander, and H. Kälviäinen, "Cost reduction and quality improvement in software testing," in *Proceedings of the 14th International Software Quality Management Conference (SQM '06)*, Southampton, UK, April 2006.

[13] O. Taipale, K. Smolander, and H. Kälviäinen, "Factors affecting software testing time schedule," in *Proceedings of the Australian Software Engineering Conference (ASWEC '06)*, pp. 283–291, Sydney, Australia, April 2006.

[14] O. Taipale, K. Smolander, and H. Kälviäinen, "A survey on software testing," in *Proceedings of the 6th International SPICE Conference on Software Process Improvement and Capability dEtermination (SPICE '06)*, Luxembourg, May 2006.

[15] N. C. Dalkey, *The Delphi Method: An Experimental Study of Group Opinion*, RAND, Santa Monica, Calif, USA, 1969.

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

[17] R. Torkar and S. Mankefors, "A survey on testing and reuse," in *Proceedings of IEEE International Conference on Software—Science, Technology and Engineering (SwSTE '03)*, Herzlia, Israel, November 2003.

[18] C. Ferreira and J. Cohen, "Agile systems development and stakeholder satisfaction: a South African empirical study," in *Proceedings of the Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT '08)*, pp. 48–55, Wilderness, South Africa, October 2008.

[19] J. Li, F. O. Bjørnson, R. Conradi, and V. B. Kampenes, "An empirical study of variations in COTS-based software development processes in the Norwegian IT industry," *Empirical Software Engineering*, vol. 11, no. 3, pp. 433–461, 2006.

[20] W. Chen, J. Li, J. Ma, R. Conradi, J. Ji, and C. Liu, "An empirical study on software development with open source components in the Chinese software industry," *Software Process Improvement and Practice*, vol. 13, no. 1, pp. 89–100, 2008.

[21] R. Dossani and N. Denny, "The Internet's role in offshored services: a case study of India," *ACM Transactions on Internet Technology*, vol. 7, no. 3, 2007.

[22] K. Y. Wong, "An exploratory study on knowledge management adoption in the Malaysian industry," *International Journal of Business Information Systems*, vol. 3, no. 3, pp. 272–283, 2008.

[23] J. Bach, "Test automation snake oil," in *Proceedings of the 14th International Conference and Exposition on Testing Computer Software (TCS '99)*, Washington, DC, USA, June 1999.

[24] M. Fewster, *Common Mistakes in Test Automation*, Grove Consultants, 2001.

[25] A. Hartman, M. Katara, and A. Paradkar, "Domain specific approaches to software test automation," in *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE '07)*, pp. 621–622, Dubrovnik, Croatia, September 2007.

[26] C. Persson and N. Yilmaztürk, "Establishment of automated regression testing at ABB: industrial experience report on 'avoiding the pitfalls'," in *Proceedings of the 19th International Conference on Automated Software Engineering (ASE '04)*, pp. 112–121, Linz, Austria, September 2004.

[27] M. Auguston, J. B. Michael, and M.-T. Shing, "Test automation and safety assessment in rapid systems prototyping," in *Proceedings of the 16th IEEE International Workshop on Rapid System Prototyping (RSP '05)*, pp. 188–194, Montreal, Canada, June 2005.

[28] A. Cavarra, J. Davies, T. Jeron, L. Mournier, A. Hartman, and S. Olvovsky, "Using UML for automatic test generation," in *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA '02)*, Roma, Italy, July 2002.

[29] M. Vieira, J. Leduc, R. Subramanyan, and J. Kazmeier, "Automation of GUI testing using a model-driven approach," in *Proceedings of the International Workshop on Automation of Software Testing*, pp. 9–14, Shanghai, China, May 2006.

[30] Z. Xiaochun, Z. Bo, L. Juefeng, and G. Qiu, "A test automation solution on gui functional test," in *Proceedings of the 6th IEEE International Conference on Industrial Informatics (INDIN '08)*, pp. 1413–1418, Daejeon, Korea, July 2008.

[31] D. Kreuer, "Applying test automation to type acceptance testing of telecom networks: a case study with customer participation," in *Proceedings of the 14th IEEE International Conference on Automated Software Engineering*, pp. 216–223, Cocoa Beach, Fla, USA, October 1999.

[32] W. D. Yu and G. Patil, "A workflow-based test automation framework for web based systems," in *Proceedings of the 12th IEEE Symposium on Computers and Communications (ISCC '07)*, pp. 333–339, Aveiro, Portugal, July 2007.

[33] A. Bertolino, "Software testing research: achievements, challenges, dreams," in *Proceedings of the Future of Software Engineering (FoSE '07)*, pp. 85–103, Minneapolis, Minn, USA, May 2007.

[34] M. Blackburn, R. Busser, and A. Nauman, "Why model-based test automation is different and what you should know to get started," in *Proceedings of the International Conference on Practical Software Quality*, Braunschweig, Germany, September 2004.

[35] P. Santos-Neto, R. Resende, and C. Pádua, "Requirements for information systems model-based testing," in *Proceedings of the ACM Symposium on Applied Computing*, pp. 1409–1415, Seoul, Korea, March 2007.

[36] ISO/IEC and ISO/IEC 15504-1, "Information Technology—Process Assessment—Part 1: Concepts and Vocabulary," 2002.
[37] K. M. Eisenhardt, "Building theories from case study research," *The Academy of Management Review*, vol. 14, no. 4, pp. 532–550, 1989.
[38] EU and European Commission, "The new SME definition: user guide and model declaration," 2003.
[39] G. Paré and J. J. Elam, "Using case study research to build theories of IT implementation," in *Proceedings of the IFIP TC8 WG 8.2 International Conference on Information Systems and Qualitative Research*, pp. 542–568, Chapman & Hall, Philadelphia, Pa, USA, May-June 1997.
[40] A. Strauss and J. Corbin, *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*, SAGE, Newbury Park, Calif, USA, 1990.
[41] ATLAS.ti, The Knowledge Workbench, Scientific Software Development, 2005.
[42] M. B. Miles and A. M. Huberman, *Qualitative Data Analysis*, SAGE, Thousand Oaks, Calif, USA, 1994.
[43] C. B. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 557–572, 1999.
[44] C. Robson, *Real World Research*, Blackwell, Oxford, UK, 2nd edition, 2002.
[45] N. K. Denzin, *The Research Act: A Theoretical Introduction to Sociological Methods*, McGraw-Hill, New York, NY, USA, 1978.
[46] A. Fink and J. Kosecoff, *How to Conduct Surveys: A Step-by-Step Guide*, SAGE, Beverly Hills, Calif, USA, 1985.
[47] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, et al., "Preliminary guidelines for empirical research in software engineering," *IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 721–734, 2002.
[48] T. Dybå, "An instrument for measuring the key factors of success in software process improvement," *Empirical Software Engineering*, vol. 5, no. 4, pp. 357–390, 2000.
[49] ISO/IEC and ISO/IEC 25010-2, "Software Engineering—Software product Quality Requirements and Evaluation (SQuaRE) Quality Model," 2008.
[50] Y. Baruch, "Response rate in academic studies—a comparative analysis," *Human Relations*, vol. 52, no. 4, pp. 421–438, 1999.
[51] T. Koomen and M. Pol, *Test Process Improvement: A Practical Step-by-Step Guide to Structured Testing*, Addison-Wesley, Reading, Mass, USA, 1999.
[52] P. Kruchten, *The Rational Unified Process: An Introduction*, Addison-Wesley, Reading, Mass, USA, 2nd edition, 1998.
[53] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*, Prentice-Hall, Upper Saddle River, NJ, USA, 2001.
[54] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, Reading, Mass, USA, 2000.
[55] B. Glaser and A. L. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*, Aldine, Chicago, Ill, USA, 1967.
[56] C. Kaner, "Improving the maintainability of automated test suites," *Software QA*, vol. 4, no. 4, 1997.
[57] D. J. Mosley and B. A. Posey, *Just Enough Software Test Automation*, Prentice-Hall, Upper Saddle River, NJ, USA, 2002.
[58] D. Foray, *Economics of Knowledge*, MIT Press, Cambridge, Mass, USA, 2004.