Accounting Testing in Software Cost Estimation: A Case Study of the Current Practice and Impacts

Jurka Rahikkala[1,2], Sami Hyrynsalmi[2], Ville Leppänen[2]

[1] Vaadin Ltd, Turku, Finland
jurka.rahikkala@vaadin.com
[2] Department of Information Technology, University of Turku, Finland
{sthyry, ville.leppanen}@utu.fi

Abstract. Testing has become an integral part of most software projects. It accounts for even as high a share as 40% of the overall work effort. At the same time software projects are systematically exceeding their effort and schedule forecasts. Regardless of the overruns and big share of testing, there is very little advice for estimating testing activities. This case study research assesses the current practice of estimating testing activities, and the impact of these practices on estimation and project success. Based on the interviews with 11 stakeholders involved in two case projects and examination of project documentation, this study shows that companies easily deviate from their standard procedures, when estimating testing. This may even lead to severe estimation errors. The deviations can be explained by negative attitudes towards testing. Furthermore, this study shows that the extant literature has sparsely addressed estimation of software testing.

Keywords: Software Cost Estimation, Software testing, Project Management

1 Introduction

Software cost estimation (SCE), or effort estimation, is an art which is not well handled by the software industry (see e.g. Rahikkala et al., 2014). The famous *Chaos Reports* by Standish Group International have for decades claimed that nearly two thirds of software products either failed to meet their budget and timetable goals or they were never delivered (The CHAOS Manifesto, 2013). While these numbers have been heavily criticized (see Laurenz Eveleens and Verhoef, 2010), high failure percentages have been shown also by other research and consultancy institutes (c.f. Moløkken and Jørgensen, 2003; Mieritz, 2012). Nevertheless, these numbers clearly show the dilemma of software cost and effort estimations: they often fail.

In this study, our topic is an increasingly important aspect of SCE: estimating the effort of software testing activities. The software testing activities have been found to take from 20%–25% (Haberl et al., 2012; Lee et al., 2013; Buenen and Teje, 2014) to even 40% or more (Ng et al., 2004) of the total cost of the project. However, there is

still a lack of knowledge of SCE methods as well as proper tools and practices for software testing estimation. For example, from the studies included into Jørgensen and Shepperd's (2007) systematic literature review on software cost estimation, only a handful seems to discuss estimation of software testing activities.

Thus, there is a gap in extant literature on the effect of software testing effort estimation. The objective of this paper is to 1) gain in-depth knowledge of the current practice for accounting testing in SCE, and 2) understand the impact of the used practices on SCE and project success.

To answer the research questions, a qualitative research approach is used. We study software cost estimation practices in two projects in two case companies. Based on the study of 11 interviews and 17 project related documents, this paper contributes to the scientific literature by reporting on the current practice of estimating testing effort, and the impact of use of these practices on estimation and project success. Understanding the role of testing effort estimation in software projects better may help project managers, other software professionals and researchers to pay more attention on testing and related estimation.

The rest of this paper is structured as follows. Section 2 briefly presents the background of software cost estimation studies. It is followed by a description of research design and the case study subjects. The fourth section presents the results and Section 5 discusses the practical and scientific implications, and addresses the limitations of the study as well as the future research directions. The final section concludes the study.

2 Background

Software cost and effort estimation has a long tradition in the field of computing disciplines. For example, Farr and Nanus (1964) and Nelson (1966) addressed the cost estimation of programming five decades ago. Since then a multitude of software cost estimation methods and models have been presented (c.f. Boehm et al., 2000; Briand and Wieczorek, 2002; Jørgensen and Shepperd, 2007). However, as discussed in the introduction, the software industry has an infamous history with the success of software cost and effort estimations. Despite the decades of research, projects often run over their budgets and timetables (Moløkken and Jørgensen, 2003).

Software testing is showed to be hard to estimate and predict. In the survey of Australian companies by Ng et al. (2004), only 11 companies out of the 65 studied were able to meet their testing budget estimates. Most of the companies spent more than 1.5 times the resources they had estimated. Furthermore, three of the studied companies estimated the cost of testing twice as high as was the actual cost. At the same time, most of the companies reported using approximately 10-30% of their initial budget to the testing activities.

The extant literature has classified presented estimation methods in several ways. For example, Boehm, Abts, and Chulani (2000) divided the existing methods into six different categories: Model-based, Expertise-based, Learning-oriented, Dynamics -based, Regression-based and Composite techniques. However, for the sake of simplicity, we consider only two major types of software cost estimation methods: Model-based and Non-model-based (Briand and Wieczorek, 2002). The first category, in general, consists of methods that have a model, a modelling method, and an application method. A classical example of these kinds of software cost estimation methods is COCOMO by Boehm (1981). The methods of the second category consist of one or more estimation techniques. These methods do not involve any models, only estimations. For example, bottom-up expert estimations belong to this category.

Some methods for software test effort estimation have been presented. For example, Calzolari, Tonella and Antoniol (1998) presented a dynamic model for forecasting the effort of testing and maintenance. The model is based on a view that testing is a similar activity to predating a prey (i.e. software bug) in nature. Engel and Last (2007) have also presented a model for estimating testing effort based on fuzzy logic. In contrast to modelling-focused techniques, Benton (2009) recently presented the IARM-estimation method for software cost and effort. The model is remarkably simpler than presented formal models and it emphasizes expert evaluation by team members; however, there is a lack of validation and verification of the proposed model.

To summarize, software testing activities seem to be rather hard to estimate. While there is a lack of studies on software testing estimations, Ng et al. (2004) showed that most of surveyed Australian companies do not hit their estimates. Furthermore, there is a rather wide understanding on the size of software testing of the whole software development project. While some studies suggest the use of only one-fourth of the budget (e.g., Haberl et al., 2012; Lee et al., 2013; Buenen and Teje, 2014), there have been claims of even half of the budget (see e.g. Ammann and Offutt, 2001). Nevertheless, in our literature review, we sparsely found any studies focusing on developing or validating software cost estimation techniques for testing. This is a noteworthy disparity between the cost caused by testing activities and the academic interest towards the topic.

3 Research methodology

In the following, we will first present the research approach and design of this study. It is followed by a description of case study subjects.

3.1 Research design

This study is based on a qualitative research approach (Cresswell, 2003). We use a case study research strategy and interviews as the main tools of inquiry. The qualita-

tive research approach was selected to allow us to get an in-depth understanding about the phenomenon under the study lens. The case study research strategy was used as the researchers have no control over the study subject (Yin, 2003). As Patton (2001) states, the case studies are able to shed light on phenomena occurring in a real-life context. This study is *exploratory* of type, finding out what is happening, seeking new ideas and generating hypotheses and ideas for new research (Robson, 2002). The research uses a multiple case study design following a replication logic (Yin, 2003). The case study organisations were selected based on the impression of the SCE maturity of the organisation gained during the initial discussions with the organisation representatives. The unit of analysis is a single software cost estimate. The study is focused on the experiences gained during the preparation of the cost estimate.

This study about estimation of software testing consists of two case companies that we call as *Small Global* and *Large Multinational*. The companies wished to remain anonymous in this study. From both companies, we selected one software implementation project that has either ended or is near its ending. The first project was developed for a customer, thus the estimate was used for pricing the project, in addition to other planning purposes. The other is a software tool development project, which is aimed for a mass market. In this project, the estimates were especially used for estimating the release date of the product.

For the selected case study projects, we interviewed different stakeholders involved in the projects. The interviewees' roles varied from developers and testers to project managers and senior executives. In addition to the interviews, we collected various documents related to the project. For example, we reviewed project plans, design documents and minutes of weekly meetings related to the projects.

We created an interview protocol consisting of several questions related to software cost estimation and testing. The protocol was created following the guidelines by Runeson et al. (2012). The interviews were conducted as semi-structured (Robson, 2002). In the interviews, while following the protocol, new ideas were allowed to be brought up and discussed with the interviewee. The interviews were conducted by two researchers where one acted as the main interviewer. An interview session was approximately one hour in length and the discussion was recorded. The recordings were transcripted and sent to the interviewees for review. All case subjects participated in the study voluntarily and anonymously, and the collected data was treated as confidential.

For the analysis of data, we used nVivo 10. All transcripted interviews, notes done during the interviews, in addition to the auxiliary materials, were imported into the software. The analysis was conducted in a series of steps (Robson, 2002). First the texts were coded by the researchers, whereafter iteration followed, until the conclusions were reached.

3.2 Case subjects

The Small Global is a software producing firm of about 100 persons, headquartered in Finland. The company's line of business consists of selling consultancy and support services in addition to software products to businesses. The company is global; it has customers and offices in several countries. The selected project, referred to as *Developer Tool*, is a typical software product development project in the company. In the project, the aim was to produce a visual design tool for developing applications. The project followed a Waterfall-style software engineering method: it was strictly planned up-front but the actual development work was divided into sprints. The interviews conducted in the company are shown in Table 1.

The Developer Tool project started with a prototype where technical challenges and possible development stacks were studied. After the prototype project, a project aiming at the release of version 1.0 was planned. The management named a project manager and a product owner for the product. The product owner crafted a design document for the product, and based on that document, the project management created a project plan with time estimates. Initially, the project was estimated to take three months with a team of four people.

The project missed its first deadline, a beta version release, approximately after the half-point of the estimated duration of the project. First, the project team narrowed the content plan of the product in order to hit the planned release date. Later, the initial content was returned to the plan and deadlines were postponed in the future. When the problems were noted, some developers were changed and new ones were introduced. Currently, the project is expected to finish after approximately nine months of development with the team of approximately four persons.

Table 1. Interviews done in this research

Interview code	The role of interviewee
	Small Global
Interview V_1	Key informant interview (Product owner)
Interview V_2	Senior manager of *Developer Tool*
Interview V_3	Product owner of *Developer Tool*
Interview V_4	Senior manager of *Developer Tool*
Interview V_5	Project manager of *Developer Tool*
	Large Multinational
Interview T_1	Key informant interview (Project manager)
Interview T_2	Project manager of *Operational Control System*
Interview T_3	Senior manager of *Operational Control System*
Interview T_4	Software test manager of *Operational Control System*
Interview T_5	Requirements expert of *Operational Control System*
Interview T_6	Senior developer of *Operational Control System*

The other case study subject comes from The Large Multinational, which is a large multinational company also headquartered in Finland. The Large Multinational produces software and consultancy for a wide area of business sectors. The selected project, referred to as *Operational Control System*, is a typical software development project for the company. The resulted software is a business intelligence reporting system for following certain control activities. The software was ordered by a long-term customer of the company. The interviews conducted in the company are shown in Table 1.

Also this project followed a Waterfall-like software development process: the requirements were elicited before the features of the product were agreed upon with the customer. The product was estimated only after the specification documentation was ready and accepted by the customer. The estimation was done by the developers and testers themselves. The Large Multinational uses a structured sheet for estimations that the workers filled individually by themselves. The project manager prepared the final estimates by using the expert estimates as the base.

The Operational Control System project was planned according to certain preconditions: the customer had a fixed budget and limited time for development. Only after both the customer and the software vendor had agreed upon the content with a limited set of unknown features, the development started. The development work continued rather straightforwardly from design and implementation to testing and delivery. The project lasted 10 months; the size of the project was approximately 30 man-months, of which 25% was used for testing and quality assurance. While there were major changes asked by the customer in the midway of the project, it hit the targets by keeping the budget and the timetable. In certain areas, there were estimation mismatches: in one software testing area, there was a significant overrun (+76% of initial estimate); however, in the implementation of a certain feature, an expert level developer was able to underbid the estimate (-77%) that was created with the presumption of a general level developer.

4 Results

This section presents the findings identified during the analysis of data, as described in the research methodology chapter. The findings are grouped into the following five categories:

1. General
2. Estimation practices
3. Testing plan
4. Attitudes
5. Estimation challenges

4.1 General

In both projects testing was strongly related to effort and schedule overruns. In the Operational Control System project the overrun of the data import testing was nearly 80%, although the overall project managed to keep the original budgets. In the Developer Tool project, omitted testing tasks resulted in significant effort and schedule overruns, being roughly 100% in July 2015. Regardless of the significant testing related overruns, the management in both projects report that the actual amount of work would probably not have been affected by more accurate estimates. In other words, the actual testing work was necessary, although bigger than anticipated.

The overrun of the testing effort in the Operational Control System lead to discussions with the customer, and likely to decreased customer satisfaction. In the Developer Tool, the project has received a late project status because of the overruns. This has lead to some negative late project symptoms, like decreased development team motivation, increased number of status meetings, frequent re-estimation and more discussions about the project scope (McConnell, 2006).

Both organisations have a long experience of software projects and related testing. The unit responsible for the Operational Control System has dedicated testing engineers and teams, while in the Developer Tool project the testing was conducted partially by dedicated testing engineers, partially by the development team. In both cases customer representatives or pilot users participated in user testing. Following the Testing Maturity Model (TMM) presented by Burstein et al. (1998), the testing maturity was on level 3 and level 2 in Operational Control System and Developer Tool, respectively, in the scale from 1 (low maturity) to 5 (high maturity).

4.2 Estimation practices

There were standard procedures for the estimation of the testing effort in both of the companies. The procedures required that the project plan, functional specification and testing plan must be ready before the effort estimation. Expert estimation was used for estimating testing tasks in both projects, i.e. experts estimated the effort of the testing tasks based on their professional competence. In the Operational Control System project a spreadsheet based work breakdown structure was used for preparing the final estimate, and the effort was also compared with the historical project data. There was also a guideline based on the historical data that the testing effort varies between 50% and 200% of implementation effort, depending on the application area to be tested. The expert estimation of testing tasks was conducted by the actual testing engineer in the Operational Control System project and by the product owner in the Developer Tool, while feature implementation tasks were estimated by the actual software engineers in both projects. In the Operational Control System, the actual testing engineers

were known by the time of estimation, which was not the case in the Developer Tool project. Finally the estimates were reviewed in the project steering groups.

The fact that the actual testing engineer estimated the testing tasks was seen to have a positive impact on the estimation results in the Operational Control System project, as well as the experience of the estimator. The Operational Control System team members report also that knowing the actual persons who will be doing the testing helps in preparing estimates. That is, the amount of work hours needed depends on the persons doing the work.

4.3 Testing Plan

Regardless of the requirement by the standard estimation procedures, neither of the projects had a testing plan ready, when the implementation phase of the project was started. In other words, the exact scope of the testing was not defined in detail before testing was estimated. In the Operational Control System project the testing effort was accounted in the project plan, and the testing manager reports that there was a shared vision of the testing based on the previous projects. In the Developer Tool project only automated regression testing was accounted in the project plan, but user testing and manual testing were omitted. In both projects the testing plan was created in a later phase of the project. The reason for not finalizing the testing plan before starting the implementation was a hurry to proceed with the implementation, not to save money.

Interviewees in the Developer Tool report that there was a significant difference in the expectations of the project results between the development team and the management. The management expected a finalized, user tested product, while the development team's expectation was more like a regression tested proof-of-concept, where the user testing would have been postponed until the next version of the product. The overruns in the Developer Tool project are specifically related to this difference in expectations. Although the testing activities are seen as necessary, a senior manager in the Developer Tool states that a testing plan would have helped to discover testing related problems earlier and to plan testing activities better. A proof-of-concept project was done prior to the actual Developer Tool project to test all key technological assumptions, but that did not cover testing. Retrospectively a senior manager speculates that testing should have been part of the proof-of-concept to avoid testing related surprises.

In the Operational Control System the project manager reported that the first estimate for testing did not fit in the project schedule, and it was discovered that the estimate was prepared in five minutes. The project manager had also challenged the high effort for testing activities, being not able to understand the basis for the estimate. Furthermore, the project and testing managers report that the difficulties in testing

were related to e.g. unavailability of test data and higher complexity of the data import logic than expected.

4.4 Attitudes

In the Operational Control System, the testing manager describes that testing is generally considered as of low importance, and that the competence and historical data is not valued as it is valued in the implementation. This feeling contradicts with the general comments where professional competence and use of historical data are strongly connected to estimation success. The Operational Control System project owner emphasizes that the testing effort must be on a reasonable level considering the application area and that it must add value to the project. The project owner continues that Large Multinational has invested significant amount of time and money for improving testing capabilities within the past two years. This communicates about the experienced importance of testing. However, according the product owner, changing established practices will take time. Based on the interviews, changing attitudes seem to take time, like changing any other capabilities. In both projects the technical staff considered especially automated regression testing to have a high importance. In the Developer Tool project, a senior manager describes that developers' attitudes towards manual testing and user experience testing are negative.

Other findings in this research support the feeling that testing is not considered equally important as implementation. For example, the testing plans were not ready when the project was estimated or testing started, and in one project the actual testers did not estimate the testing work or the actual testers were not known at the time of estimation. The Developer Tool projects's product owner reports that testing related tasks are first omitted if the schedule is tight and something needs to be dropped out from the scope. This lower level of importance may prevent testing as a function from evolving, including estimation of testing. The testing manager in the Operational Control System pointed out that the attitudes towards testing are not encouraging even in universities: *"There was only one course of software testing out of two hundred offered"*. Additionally, the testing manager commented that testing is seen as a task for those who are not good enough for ordinary programming.

4.5 Estimation challenges

In the Operational Control System project, the testing manager considered estimation of testing as more difficult than estimating implementation. The primary reason for this is the high number of dependencies: problems with data or specifications, or high number of bugs will reflect to testing. For example, if the test data is delayed by one week, the testing team is still there, but not doing what they were planned to do. The number of found bugs was relatively high in the Operational Control System, which

cumulated extra work because of manual testing practices. Also the Developer Tool project manager emphasised the role of external parties in estimation. For example, the unavailability of external testing engineers or users for user testing may delay the project. Other interviewees could not make a difference in difficulty, but a senior manager in the Developer Tool pointed out that there is generally less experience of estimating testing.

Automated testing was widely seen to make regression testing more predictable, because then the regression would not cause additional testing work. The project manager in the Developer Tool emphasises the importance of the right degree and phase of testing, otherwise the maintenance of tests can generate extra work. A senior executive of the project continues that indecisiveness in processing user feedback may cause overruns, and underlines the importance of decisiveness and time boxing in user testing.

5 Discussion

This section discusses the main case study findings, and presents the related practical and theoretical implications. This section also addresses the limitations of this study, and gives pointers for further research.

This study has captured in-depth experiences from two *typical* project organisations, who have established software development practices for one project, based on their internal guidelines. As is typical for these kinds of organisations, the maturity and optimisations of the practices are not on an exemplary level, because the projects exist for only a certain period of time, and the contents of the projects vary. The primary need of these organisations is to get the basic things right in every project, not to optimise the last details.

5.1 Implications for practice

This study clearly shows that the basic principles for estimating testing related tasks are the same as for estimating implementation tasks, but there is a strong tendency to take shortcuts and postpone tasks beyond their planned deadlines. The most significant finding is that the testing plan was not delivered in either project by the time of estimation. This can be compared to estimation of implementation with incomplete or no specifications, which is connected to overruns (Standish group, 1994; Lederer and Prasad, 1995). Especially in the Developer Tool project this was the reason for overruns, and the evidence suggests also that some surprises could have been avoided also in the Operational Control System project with a finalised testing plan.

Also, the practical estimation tasks were subject to taking short-cuts in the Developer Tool. While the implementation tasks were estimated by a developer and the initial project team was known, the testing was estimated by the product owner for an

unknown team. This is problematic from the estimation point of view, because the productivity of developers is known to vary significantly (Kemayel, Mili and Ouederni, 1991).

The attitudes towards testing and its estimation seem to be contradictory: on one hand testing cannot be omitted, but on the other hand it seems not to be a full-fledged part of the software project. This research has found that the testing personnel perceives that testing is considered as unimportant, and the management emphasises that the cost needs to be reasonable. The tendency to take shortcuts supports this perceived unimportance. The extant literature clearly shows that the attitudes of the senior management influences strongly the change or improvement of a certain area (Stelzer and Mellis, 1998; Wiegers, 1998), which means that the negative attitudes are likely to influence also the estimation of testing tasks negatively.

The technical staff in both projects has very positive attitudes towards automated testing. This is claimed to reduce manual work and improve predictability. This is a fair assumption, and confirmed, for example, by Ramler and Wolfmeier (2006) and Hoffman (1999), but only if implemented properly. They emphasise that the project characteristics need to be accounted when planning testing, in order to implement effective and reasonably priced testing. This was supported by the comments from a senior manager in the Operational Control System (*"the extent of testing must consider the project at hand"*) and (*"testing must add value to the project"*), the project manager from the Developer Tool (*"degree and phase of testing must be carefully planned"*) and a senior manager in the Developer Tool (*"decisiveness and time boxing is important in user testing"*).

Changing attitudes, as well as other capabilities, seem to take time. The Large Multinational has invested resources in building capabilities significantly within the past two years, but the know-how or attitudes are still not on the same level as in implementation. This corresponds to results from other change situations (Piderit, 2000).

As a summary, similar projects as the Developer Tool and Operational Control System are recommended to follow the same estimation practices for testing related tasks as for implementation. Neither of the projects reported that the reason for postponing or omitting tasks was cutting cost. In this sense, it seems counter intuitive and productive not to follow the same rigour in testing as with implementation, especially when the short-cuts seem to cause even severe estimation errors.

Finally, it was noted in the interviews that the personnel was rarely properly educated to use the estimation tools. While the other of our case study companies offered support and education in the estimation to the project management, the development teams were not aware of these services. Furthermore, only a few mentioned university education as a source of cost estimation knowledge, even though cost estimation based on an expert's opinion is nowadays a part of the standard workflow in many companies. While effort estimation is a part of the curriculum guidelines suggested by

ACM[1], there might be a need for education organisations to re-evaluate the content of teaching.

5.2 Implications for theory

The current SCE literature provides only a few case studies providing in-depth knowledge of real-life situations (Jørgensen and Shepperd, 2007), and, to our best knowledge, this is among the first studies to report on experiences related to estimating software testing. This paper contributes to the body of knowledge by showing that organisations seem to deviate from their standard practices when estimating testing related tasks, causing estimation errors. The reason for deviations resides in negative attitudes towards testing. Cost savings are rejected as a reason for deviations, as well as poorly performed testing as a reason for overruns in testing.

Furthermore, our literature review shows that there is a lack of empirical and theoretical studies on addressing software testing effort estimation. While further work such as a systematic literature review is needed to confirm this observation, this hints that one of the reasons for software cost estimation related problems is the lack of proper focus on testing and its effort estimation. However, new estimation tools or methods for software testing are not silver bullets. A holistic view is needed for improving the accuracy of software estimates.

5.3 Limitations and further research

This study has certain limitations. First, the findings of this study are subject to constraints of the research methodology. This research studied two very similar software projects, which limits the validity of the findings to similar contexts. It is recommended that further research would be conducted in a different context to see if e.g. larger projects or continuous product development projects suffer from similar testing estimation related problems as reported in this paper. Second, a qualitative study is always, to some degree, subjective and might be influenced by the expectations and opinions of the researchers and interviewees. However, we did our best to treat the data objectively and tool countermeasures to reduce bias. For example, all transcripts and quotes, as well as the manuscript, were checked and approved by the interviewees before the publication.

Considering the big share of testing work in the overall software project and the problems reported in this study, further research of the topic is justified. First, our unstructured literature review did not reveal many studies in this topic. Thus, a more thorough literature review should focus to shed more light on possible research gaps

[1] Software Engineering 2014. Curriculum Guidelines for Undergraduate Degree in Software Engineering. https://www.acm.org/education/SE2014-20150223_draft.pdf

in software testing estimation. Second, there is a lack of studies addressing the methods and tools used in the industry to estimate software testing. Further work should be devoted to reveal the best practices already used in the industry.

6 Conclusions

This research provides evidence that software testing related tasks in software projects are estimated by using similar practices as for implementation related tasks, but deviations from the standard practice occur often. These deviations, such as estimating without a testing plan, estimating work that will be carried out by others or estimating without knowing the actual testers, have been a source for even severe overruns. The research rejects poorly performed testing as a source for overruns. Overruns themselves are a source of decreased team motivation and customer satisfaction, and additional work, among other things.

The results of this research suggest that the reason for process deviations resides in the negative attitudes towards testing. Widespread deviations from established practices among both project management and technical staff, together with the direct comments from the interviews, indicate that testing is not considered as important as implementation, and therefore deviations are allowed. The results reject cost savings as the reason for deviations.

The main implications from the results for software managers, experts, project managers and academia are the following:

- A deviating estimation process for software testing may lead to severe estimation errors.
- Software projects should use the same rigor in estimating testing as for estimating implementation. Deviations should not be allowed.
- Managers responsible for software and project processes must recognise the importance of testing and promote the importance of it to change the attitudes towards testing. This is necessary in order to establish the use of good practices in organisations.

Finally, the aforementioned serve also as a good starting point for further research.

Acknowledgements.
The authors gratefully acknowledge Tekes – the Finnish Funding Agency for Innovation, DIGILE Oy and Need for Speed research program for their support.

References

1. Ammann, P., Offutt, J.: Introduction to Software Testing. Cambridge University Press, UK (2008)
2. Benton, B.: Model-Based Time and Cost Estimation for Software Testing Environment. In: Sixth International Conference on Information Technology: New Generations, pp. 801-806. IEEE (2009)
3. Boehm, B., Abts, C., Chulani, S.: Software development cost estimation approaches — A survey. Annals of Software Engineering, 10(1-4), 177-205 (2000)
4. Boehm, B.: Software Engineering Economics. Prentice-Hall, Englewood Cliffs, New Jersey (1981)
5. Briand, L.C., Wieczorek, I.: Resource Estimation in Software Engineering. Encyclopedia of Software Engineering. John Wiley & Sons, Inc., New Jersey (2002)
6. Buenen, M., Teje, M.: World Quality Report 2014-15: Sixth Edition. Capgemini, Sogeti and HP (2014)
7. Calzolari, F., Tonella, P., Antoniol, G.: Dynamic model for maintenance and testing effort. In: Proceedings of the International Conference on Software Maintenance 1998, pp. 104-112. IEEE (1998)
8. Creswell, J.: Research Design: Qualitative and Quantitative and Mixed Methods Approaches. Second edition. SAGE Publications, Inc., Thousand Oaks, California (2003)
9. Engel, A. Last, M.: Modelling software testing costs and risks using fuzzy logic paradigm. Journal of Systems and Software, 80(6), 817-835 (2007)
10. Farr, L., Nanus, B.: Factors that Affect the Cost of Computer Programming. Volume I. Technical Documentary Report No. ESD-TDR-64-448. United States Air Force, Bedford, Massachusetts (1964)
11. Haberl, P., Spillner, A., Vosseberg, K., Winter, M.: Survey 2011: Software Testing in Practice. Auflage, dpunkt.verlag, (2012) http://www.istqb.org/documents/Survey_GTB.pdf
12. Hoffman, D.: Cost Benefits Analysis of Test Automation, Software Testing Analysis & Review Conference (STAR East). Orlando, FL (1999)
13. Jørgensen, M., Shepperd, M.: A Systematic Review of Software Development Cost Estimation Studies. IEEE Transaction on Software Engineering, 33(1), 33–53 (2007)
14. Kemayel, L., Mili, A., Ouederni, I.: Controllable factors for programmer productivity: A statistical study. Journal of Systems and Software 16(2), 151-163 (1991)
15. Laurenz Eveleens, J. and Verhoef, C.:. The rise and fall of the Chaos Report figures. IEEE Software, 27(1), 30–36 (2010)
16. Lederer, A.L., Prasad, J.: Causes of Inaccurate Software Development Cost Estimates. Journal of Systems and Software, 31(2), 125-134 (1995)
17. Lee, J., Kang, S., Lee, D.: Survey on software testing practices. IET Software, 6(3), 275-282 (2012)
18. Mieritz, L.: Surveys Shows Why Projects Fail. G00231952. Gartner, Inc. (2012)
19. Moløkken, K., Jørgensen, M.: A Review of Software Surveys on Software Effort Estimation. In: Proceedings of International Symposium on Empirical Software Engineering, pp. 220–230. IEEE (2003)
20. Nelson, E. A.: Management Handbook for the Estimation of Computer Programming Costs. Technical Documentary Report No. ESD-TR-67-66. United States Air Force, Bedford, Massachusetts (1966)
21. Ng, S.P., Murnane, T., Reed, K., Grant, D., Chen, T.Y.: A preliminary survey on software testing practices in Australia, In: Proceedings of the 2004 Australian Software Engineering Conference, pp.116-125. IEEE (2004)

22. Patton, M.: Qualitative Research and Evaluation Method, Third edition. SAGE Publications, Inc., Thousand Oaks, California (2001)
23. Piderit, S.K: Rethinking Resistance and Recognizing Ambivalence: A Multidimensional View of Attitudes toward an Organizational Change. Academy of Management Review, 25(4), 753-794 (2000)
24. Rahikkala, J., Leppänen V., Ruohonen, J., Holvitie, J.: Top management support in software cost estimation: A study of attitudes and practice in Finland. International Journal of Management Projects in Business, 8(3), 513-532 (2015)
25. Ramler, R., Wolfmaier, K.: Economic perspectives in test automation: balancing automated and manual testing with opportunity cost. Proceedings of the 2006 international workshop on Automation of software test, pp. 85-91. ACM (2006)
26. Runeson, P., Höst, M., Rainer, A., Regnell, B.: Case Study Research in Software Engineering: Guidelines and Examples. John Wiley & Sons, Inc., New Jersey (2012)
27. Standish Group: The Chaos Report. The Standish Group (1994)
28. Stelzer, D., Mellis, W.: Success factors of organizational change in software process improvement. Software Process: Improvement and Practice 4(4), 227-250 (1998)
29. The CHAOS Manifesto: Think Big, Act Small. The Standish Group International (2013)
30. Wiegers, K. E.: Software process improvement: Eight traps to avoid. CrossTalk, The Journal of Defense Software Engineering (1998)
31. Yin, R. K.: Case Study Research: Design and Methods. Third edition. SAGE Publications, Inc., Thousands Oaks, California (2003)