Indirect Effects in Evidential Assessment: A Case Study on Regression Test Technology Adoption

Engström, Emelie; Feldt, Robert; Torkar, Richard

Published in:
EAST '12 Proceedings of the 2nd international workshop on Evidential assessment of software technologies

DOI:
10.1145/2372233.2372239

2012

Indirect Effects in Evidential Assessment: A Case Study on Regression Test Technology Adoption

Emelie Engström
Software Engineering Research Group
Dept. of Comp. Science
Lund University, Sweden
emelie.engstrom@cs.lth.se

Robert Feldt
Div. of Software Engineering
Chalmers University of Technology and Gothenburg University, Sweden
robert.feldt@chalmers.se

Richard Torkar
Div. of Software Engineering
Chalmers University of Technology and Gothenburg University, Sweden
richard.torkar@chalmers.se

ABSTRACT

Background: There is a need for efficient regression testing in most software development organizations. Often the proposed solutions involve automation. However, despite this being a well researched area, research results are rarely applied in industrial practice. *Aim*: In this paper we aim to bridge the gap between research and practice by providing examples of how evidence-based regression testing approaches can be adopted in industry. We also discuss challenges for the research community. *Method*: An industrial case study was carried out to evaluate the possibility to improve regression testing at Sony Ericsson Mobile Communications. We analyse the procedure undertaken based on frameworks from the evidence based software engineering, EBSE, paradigm (with a focus on the evidence) and automation literature (with a focus on the practical effects). *Results*: Our results pinpoint the need for systematic approaches when introducing a new tool. Practitioners and researchers need congruent guidelines supporting the appraisal of both the evidence base and the pragmatic effects, both direct but also indirect, of the changes. This is illustrated by the introduction of the automation perspective.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Evidence-based

Keywords

Technology Adoption, Regression Testing, Case Study

1. INTRODUCTION

The cost for testing software increases with increasing size and complexity of software systems. At the same time more efficient software development strategies have emerged which demand more frequent testing, e.g. agile strategies with continuous integration of software updates, or testing of a broader scope of products, e.g. software product line development [2, 18]. Thus the *share* of testing costs of the total development cost increases as well and testing easily becomes a schedule or cost bottleneck or it becomes inadequate (in worst case both). Regression testing is conducted in order to verify that changes to a system has not negatively impacted on previously functioning software. Thus efficient regression testing strategies are especially important in organizations where software is frequently updated or when a range of different products are derived from the same software base.

However, there is a gap between research and practice of regression testing. Even though several systematic approaches for both prioritization and selection of regression test cases have been proposed and evaluated in literature [7, 19], they are not widely used in industry [4]. In many cases the methods and techniques do not scale while in other cases industry practice is not mature enough to implement them.

Evidence based software engineering, EBSE, is a paradigm aiming to support adoption of new software technology and methods based on sound, empirical evidence. The core tool of EBSE is systematic literature reviews and the focus is on the identification and appraisal of evidence However, EBSE have been shown to be very challenging for non-researchers and rigorous and relevant evidence may not exist to answer their questions [13, 14].

In this paper we aim to bridge the gap between research and practice by providing examples of how EBSE can be applied in industry and pinpointing challenges for the research community based on experiences from a case study [6] carried out at Sony Ericsson Mobile communications, SEMC. We describe and analyze the procedures undertaken to find a relevant regression testing technique and to evaluate the benefits of introducing it into the testing process.

It is not enough to focus on direct evidence isolated to a technique but the context and the practical effects of the changes also need to be considered. Thus we combine guidelines on EBSE [3] by Dybå et al. with guidelines on automation with respect to human-computer interaction [11] by Parasuraman et al. as a framework for our analysis. A recent example of automation anlysis in technology adoption in software engineering is given by Borg [1]. This perspective is essential also when considering evidence for the efficacy of regression testing techniquessince it will involve automation of some sort. For each of the five steps of EBSE, the evaluation and decision tasks of the automation procedure are discussed and embodied with examples from our case study, followed by a discussion about current challenges and advice

WOODSTOCK '97 El Paso, Texas USA

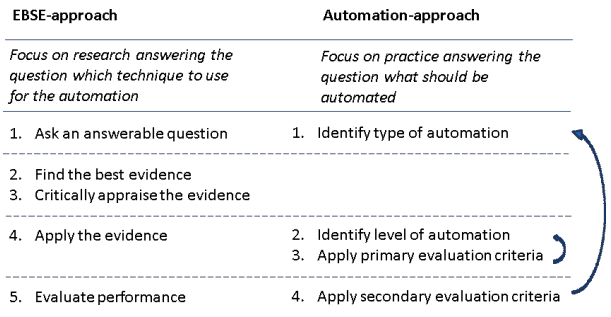

Figure 1: Overview of the EBSE approach described by Dybå et al. [3] and the automation approach described by Parasuraman et al. [11].

for practitioners and researches.

The paper is organized as follows: Section 2 describes the synthesis of the EBSE and automation model and Section 3 reports the case study and contains the analysis. Both these sections are organized according to the five different steps of EBSE. In section 4 we discuss the lessons learned about the methodological strengths and limitations of evidence-based practice and in Section 5 we conclude the paper with advice for practitioners and researchers.

2. THE ANALYSIS FRAMEWORK

Dybå et al. [3] propose a systematic EBSE approach for practitioners. It involves five steps to support and improve decisions on technology adoption. Similarly, Parasuraman et al. [11] propose a procedure involving four steps of evaluation and decision tasks that helps answer questions about what should be automated and to what level. Primary and secondary evaluation criteria are suggested to be applied iteratively while adjusting the decisions. The two models complement each other in case of introducing automation to the software engineering process. Below we describe the synthesized framework, and it's application on regression test technology adoption, according to the five steps in the EBSE approach.

2.1 Ask an answerable question.

This step corresponds to the initial steps in the Automation model answering questions on what to automate and what type of automation should take place. Parasuraman suggest a classification of automation types based on four types of human information processing. This leads to four different classes of functions that can be automated: information acquisition, information analysis, decision and action selection, and action implementation [11].

In case of test automation, the type of automation depends on the *scope* of the testing. Testing software typically involve several test activities which all may be automated in different ways with different goals: test planning, test case design, test execution and analysis of test results. Automation may seek to replace humans performing repetitive and simple tasks e.g. executing test cases or have a goal to accomplish something beyond human capabilities e.g. decision support. Regression testing research focus on the latter.

The level of testing under consideration is another factor of the automation scope. Goals and strategies for automation may vary with level of test [4]. Test planning at unit level could for example be based on code coverage analyses while at the integration testing level combinatorial strategies could be more relevant. Automation at system level is more complex and a more thorough cost benefit analysis could be a good idea.

Many challenges in regression testing are not specific for regression testing but improvements may be achieved through improvements of other tasks e.g. automation of test executions or design for testability [4]. Hence, it is important to specify the *effect targets* of the intended automation and ensure that they are aligned with the scope of the automation.

A good understanding of the context helps to identify, relevant effect targets and a proper scope for the automation. Even though there is a common understanding of what regression testing is and why it is applied in industry, the variation in practices is large. Regression testing is applied differently in different organizations, at different stages of a project, at different levels and with varying frequency [4]. Furthermore regression testing is not an isolated activity but is strongly dependent on the context in which it is applied.

2.2 Find the best evidence

The selection of technique to implement should ideally be based on research, and guided by relevant classifications of empirical evidence, typically systematic literature reviews, which could be mapped to a similarly relevant description of the current context, scope and effect target. To determine which regression testing technique is appropriate for a certain situation, several factors need to be considered such as which input does the technique require?, on what level of abstraction is the analysis made?, what are the claims of a technique, and what empirical support is there for these claims? [7]. This step has no counterpart in the Parasuraman model since they assume a different situation in which a specific type of automation has already been selected or is being evaluated. In the EBSE model one should consider the breadth of available evidence and techniques.

Several literature reviews on regression testing have been published lately. A thorough overview of different regression testing strategies in literature is given by Yoo and Harman [19] and it provides a good starting point for the search. Here the different areas of minimization, prioritization and selection are explained and investigated. A classification scheme and classification of regression test selection techniques is provided by Engstrom et al. [7].

2.3 Critically appraise the evidence

This is also a step which is specific for the EBSE approach. EBSE recommend practitioners to make use of available summaries of evidence, e.g. systematic literature reviews and systematic maps. In cases were such summaries are not available practitioners may be guided by checklists [3] in appraising (often a smaller set of) published studies.

The current evidence base on regression test selection does not offer much support for selecting the best automation technique. Although several empirical evaluations of regression testing techniques have been made lately, the evidence base is incomplete and sometimes even contradictory. Regression testing techniques are context dependent and in many cases evaluations are made only in one specific context. It is rather the different implementations of the techniques than the high level concepts that are evaluated. Moreover, often only one aspect is evaluated for a technique e.g. either

the level of cost reduction or the level of fault detection [7]. Indirect effects of introducing this type of automation are rarely discussed.

2.4 Apply the evidence

In this step the evidence should be integrated with the knowledge about the concrete situation and the pragmatic effects of the changes should be considered. Automation changes may have both direct and indirect effects. This step concerns the indirect effects , for example cognitive effects, effects on related processes (e.g. requirements engineering and design) and long term effects, and corresponds with the identification of level of automation and application of primary evaluation criteria in the Parasuraman model. Sheridan and Verplank define 10 different levels of automation [17] based on the amount of user interaction required. Although they are not defined with a focus on software testing they are applicable in this context.

The primary evaluation criteria according to Parasuraman et al focus on the consequences for the human performance after the automation has been implemented. For example, Parasuraman states that 'evidence suggests that well-designed information automation can change human operator mental workload to a level that is appropriate for the system tasks to be performed' [11]. An example can be the listing of test cases in a prioritized list to help testers make the right decision. In addition to mental workload, Parasuraman mentions situation awareness, complacency and skill degradation as primary evaluation criteria to be taken into consideration.

To help better understand the changes that an automation leads to we have found it useful to establish pre- and post-automation task flows. Detailed such analyses could involve the use of value stream maps [10] but we have found even simpler identification procedures to have value. Based on the description of the current testing task flow, and an initial decision on the technique to use for the automation (or at least the type of task to be automated a la Parasuraman), changes between current practice and improved automated practice can be identified. It is useful to consider different types of changes such as: *changed* tasks, *added* tasks and *removed* tasks. It is important not to miss any added tasks such as creation or collection of inputs that the automated process or tool requires, maintaining the automation in case of context changes or managing and processing the output from the automated process.

2.5 Evaluate performance

This step concerns the direct effects of the automation and may be evaluated against the effect targets identified in the first step. For this step the EBSE and automation guidelines are consistent. The selected technique should be evaluated within the context where it is to be applied. Results from this evaluation should be documented and may be reused in future automation projects either within the organization or as a contribution to the general knowledge base.

3. THE CASE STUDY

The case study was carried out in four steps [6] starting with A) exploratory semi-structured interviews to better understand the context, effect targets and scope of the automation. B) Select and implement a suitable method. The methods were C) quantitatively evaluated with respect to their fault detection efficiency, and finally D) the testers' opinions about the implemented methods were collected and analyzed. Results from this case study has previously been reported by Engstrom et al. [6]. In this section we describe our application of the framework in Section 2 and discuss current challenges and advices for practitioners and researches.

3.1 Asking an answerable question.

This step involves describing the context and identifying the effect targets of the regression test automation.

3.1.1 describing the context

To gain more insights into the problem we carried out interviews with a number of key persons and used a framework proposed by Petersen and Wohlin [12] to structure the context information. The context description served several purposes, identification of context constraints relevant for the selection of technique, support in the evaluation of indirect effects of changes as well as support in the communication of the results of the evaluation of direct effects through the enabling of analytical generalization. Details about the context description are reported in [6]

3.1.2 identifying the effect target

One of the problems with the current method in our case study was its dependence on experienced testers with knowledge about the system and the test cases. There was a risk that the selected test suite was either too extensive or too narrow; a tester with lack of experience in the area could have trouble estimating the required time and resources. Moreover, the selected test suite could be inefficient and misleading. Even experienced testers could select inefficient test suites since test cases were selected in a routinely manner by just selecting the same test cases for every regression test session. Since the selection was based on judgment, there was no evidence that it was the most efficient test suite. Hence, the following were the expected benefits of a tool supported selection procedure:

- increased transparency
- improved cost estimation
- increased test efficiency
- increased confidence

The scope and effect targets identified in our case study correlate with general needs in industry [4] and is partly inline with the scope of most regression test selection and prioritization techniques in literature [19]. Similar problems were identified in a survey on regression testing practices [4]. Participants found it hard to assess the impact of changes on existing code and to make good selections, to prioritize test cases with respect to product risks and fault detection ability, and to be confident in not missing safety critical faults. Determining the required amount of tests was also considered a problem as well as to assess the test coverage.

3.2 Finding the best evidence

The selected scope of the automation directed the search towards the regression testing literature, and the list of effect targets and context constraints influenced the selection of automation technique. The hypothesis that history-based

test case prioritization could improve the current situation was a starting point for the case study which to some extent made the search and selection of technique biased. Only history-based regression testing techniques which did not require source code access were considered.

Most of the proposed techniques in literature are code based (source code, intermediate code or binary code) and based on analysis at a low level of abstraction (e.g. statements). However there are examples of techniques based on analysis at higher abstraction levels as well as on other types of input. Some techniques are based on a certain type of specifications or models. There are also some recent examples of both selection and prioritization techniques based on project data, such as failure reports or execution history of a test case. One group of techniques is claimed to be safe, meaning they do not exclude any test case that have a possibility to execute parts of the system that may have been affected by a change. This property is only relevant for selection, but might be important for certain types of systems (e.g. safety critical) [7].

3.3 Appraising the evidence

In our case the selection of technique was not guided by empirical evidence on the performance of the technique. Instead the automation choice was guided by the reports of the nature of the proposed techniques. The technique proposed by Fazlalizadeh et al. [8] was selected because it was based on historic project data, could be implemented without access to source code and allowed for multiple prioritization criteria.

3.4 Applying the evidence

Our case study spans one iteration over the steps described in 2, identification of level of automation, identification of changes and evaluation of the effect of changes.

3.4.1 Identification of level of automation

In our case the automation level would be 4: *"Computer offers a restricted set of alternatives and suggests one, but human still makes and implements final decision"* according to Sheridan and Verplank [17]. Since the scope of the automation in the case study is pure decision support and not execution of tests, it corresponds to a low level of automation. On the other hand there are differences in levels within this scope that could be interesting to pinpoint. Instead of calculating total priority values by combining several prioritization criterion, priority values could be calculated for each criteria leaving the weighting of importance to the human which would be an example of automation at even lower level but still not zero.

3.4.2 Identification of changes

The scope of the automation in our case study involved the test planning and selection of test suites for regression testing. No common procedure for how this is done existed, but different testers developed their own strategies. Thus the task flow directly affected by the automation cannot be described in more detail. This situation seems to be common in industry. [4]

Changed, added and removed tasks in the case study may be described as follows:

Changed tasks.

- The suggested automation would change the selection task from being an ad-hoc task to involve a more systematic procedure. Decisions on focus, scope and constraints of a test session are needed as input to the tool and thus have to be explicit.

Added tasks.

- New tasks relate to the use of the tool. There is an initial learning effort for using the tool. It also has to be maintained and evolve as the prerequisites change. The outcome of the tool (a selected test suite) needs to be reviewed before test execution. In this case the tool was a plug in to the supporting tool already in use at the company and it collected data from the existing test data base. No input data of any other format is required and thus no such extra tasks (development and maintenance of new artifacts) are added.

Removed tasks.

- The selection and documentation of test scope are achieved by the tool instead of manual procedures.

3.4.3 Evaluation of the effect of changes

Parasuraman's list of cognitive effects may be used as a checklist for one type of indirect effects. Another checklist may be created by combining the list of task changes with the previously described elements of the context into a matrix. Each item in the matrix represents a combination of a task change and a context facet to be considered. The indirect effects discussed in this section are identified with the help of both these checklists.

The mental effort for the decisions may increase since in many cases these types of decisions are made ad-hoc in a routinely manner. With a semi-automatic tool testers are forced to perform the selections more systematically and of course to learn and handle a tool. This increased effort might lead to that the tool is not used properly especially since the regression testing activity is so frequent. On the other hand they do not have to browse through large amounts of poorly structured test cases.

There is also *a risk in trusting the tool too much.* Decisions are complex and context dependent and the variation in regression test situations cannot be fully captured by a tool. In our case study the product maturity is low and the software system complex. The large amount of internal and hidden dependencies prevent safe selections at a reasonable cost. Furthermore, in a very changing context, historical data might not be a good predictor of fault detection probability at all. However, the quantitative evaluation in our example case showed increased efficiency in that particular case, indicating a possibility to improve regression testing with the tool. Still the suggested test suite need to be manually reviewed. Only a low level of automation is possible to achieve.

With a low level of automation the *risk of skill degradation and decreasing situation awareness* is minimal. Changing the task from manually selecting each test case to manually review a suggested test suite would probably increase the testers awareness of unexpected dependencies but also decrease their awareness of what is in the test data base and what might be missing. A more transparent regression testing approach eases synchronization of work between teams and enables their cross learning.

A positive effect of automatically generating the test plan in the test management tool instead of manually entering the test cases is *increased consistency in the documentation.* Lack of consistency in test documentation may lead to unnecessary duplication in distributed testing [5]

3.5 Evaluating performance and seek ways to improve it.

The direct effects of the implemented automation were shown to be *increased transparency* and *increased efficiency.* A prototype tool was developed for the purpose of evaluating the automation in the case study. Adaptations to the actual context were inevitable and two different versions of the technique were implemented: one as close to the original proposal as possible and one as close to the current decision procedure as possible incorporating a number of different prioritization criteria that was assumed to be important by the local experts. These criteria were identified through the initial interviews and included: historical effectiveness, execution history, static priority, age, cost, focus of test session, scope of session and current status of the test case. A more detailed description of these factors are found in [6]

Two quantitative evaluations and one qualitative evaluation were carried out to evaluate the effects of automation (Details about the evaluations are provided by Engstrom et al. [6]) with respect to the effect targets: increased transparency, increased test efficiency and increased confidence. No data regarding the *execution cost* for a test case or group of test cases was available and thus this aspect could not be evaluated.

Increased transparency is achieved with any kind of automation, since no systematic method for regression testing was currently in use. If the use of a systematic method or implemented tool does not decrease test efficiency or confidence, it is considered an improvement of the current situation.

Test efficiency regards the number of faults revealed per executed test case and was evaluated in two ways in the case study: 1) by comparing the efficiency of the execution order (prioritization) of test cases in the suites and 2) by analyzing test suite selections of the same magnitude as corresponding manually selected suites. Results from the quantitative evaluation constitutes strong evidence for the possibility to improve efficiency in the current case but little relevance in terms of possibility to generalize to other contexts.

To reach and measure *confidence* in a method is in itself non-transparent, since it deals with the gut feelings of the testers. To some extent it relates to coverage. If a method can be shown to include all presumed important test cases, the confidence in it is high. However, optimizing a method to include presumed important coverage aspects affect test efficiency negatively, since it adds test cases to the selection without respect to their probability of detecting faults.

4. DISCUSSION

In many cases adoption of software engineering evidence in practice involve some type of automation. When changing the level of automation in a process it is not enough to appraise the direct evidence of a technique since any type of automation also entails indirect effects. In our case study we identified a number of indirect effects: increased mental effort for decisions, risk for over trust in tool, skill degradation and decreased situation awareness. Also, positive indirect effects were identified such as increased situation awareness, increased consistency in test documentation, easier synchronization of work and cross learning between teams. It is not enough to answer the question "Are these evidence valid, have impact and applicable?" it is critical to consider the questions "What are indirect/additional effects when applying this and what type of evidence is there that negative indirect effects do not outweigh any positive direct effects?".

Our analysis of the regression test automation procedure show the importance of finding holistic guidelines to support the adoption of evidence, focusing on both evidence and practice. This is central for EBSE to have effect. There is a need for both general guidelines and guidelines for specific sub areas within software engineering. In our cased study we created two checklists one with automation effects to look for, based on experiences from the human-computer interaction and software test automation disciplines [11, 9], and one, pointing at changes in the context, based on the elements of the context description combined with the three types of changes: added, removed and changed tasks.

In addition to the need for methods to support the evaluation of indirect effects, our case study shows the importance of finding methods to match the communication of empirical evidence with guidelines for identifying context constraints and effect targets present in practice. This is typically an area where guidelines need to be both general and sub area specific. To provide support for practitioners in matching their context with the available evidence, frameworks must be defined at an abstraction level low enough to cover the relevant aspects of the particular group of techniques addressing their effect targets. Still the more abstract framework is needed to initially identify the scope and effect targets.

Finally, another critical issue for the success of EBSE is the existence of generalizable results. Artifacts in software engineering research tend to be context dependent, calling for systematically replicated evaluations where context factors are considered. In the case of regression test automation there is a need for evaluations of high level concepts of strategies rather than evaluations of specific implementations of techniques. To support such research it is crucial to identify the relevant context factors for a specific sub-area and define frameworks at a proper level of abstraction.

5. CONCLUSION

We have analyzed the procedure of finding a proper technique for regression test automation and evaluating the benefits of adopting it. Frameworks for EBSE and automation were combined to provide a more holistic view of the regression test automation procedure. For each of the five EBSE steps we describe the regression test automation procedure and provide examples of how to do for practitioners in similar situations. Advice for practitioners may be summarized regarding the identification of preconditions for the automation, identification of relevant research and evaluation of effects of the automation.

Identification of preconditions.

It is important to identify scope and effect targets of automation. Information may be gathered with interviews and structured according to existing frameworks [12] for context descriptions in software engineering.

Identification of relevant research.

The search for relevant techniques should be driven by scope of automation, effect targets and and context constraints. Effect targets in our case study correlate with general needs in industry and are partly addressed by research. Literature reviews on regression testing [7, 19] provide overviews of the research and are good starting points for the search. However the lack of generalizable results prevents choices based on empirics.

Evaluation of effects.

Automation changes may have both direct and indirect effects. To identify indirect effects changed, added and removed tasks need to be identified and the effects of the changes evaluated. Checklists may guide the analysis The direct effects may be evaluated against the effect targets. Frameworks for evaluating cost and efficiency of regression testing techniques are available [15, 16]

Research challenges involve 1) finding methods for evaluating indirect pragmatic effects of the automation changes as well as 2) finding methods to match the communication of the empirical evidence with the description of the industrial context where automation is under consideration.

6. ACKNOWLEDGMENTS

We thank Dr. Per Runeson for valuable review comments on this paper. The work was supported by VINNOVA (the Swedish Governmental Agency for Innovation Systems) to SWELL, Swedish research school in Verification and Validation.

7. REFERENCES

[1] M. Borg. Findability through traceability - a realistic application of candidate trace links? In *7th International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE 2012*, pages 173–181, June 2012.

[2] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, Boston, 2001.

[3] T. Dybå, B. A. Kitchenham, and M. Jorgensen. Evidence-based software engineering for practitioners. *IEEE Software*, 22(1):58 – 65, Jan. 2005.

[4] E. Engström and P. Runeson. A qualitative survey of regression testing practices. In M. Ali Babar, M. Vierimaa, and M. Oivo, editors, *Product-Focused Software Process Improvement*, volume 6156 of *Lecture Notes in Computer Science*, pages 3–16. Springer Berlin / Heidelberg, 2010.

[5] E. Engström and P. Runeson. Test overlay in an emerging software product line âĂŞ an industrial case study. *Information and Software Technology*, 2012.

[6] E. Engström, P. Runeson, and A. Ljung. Improving regression testing transparency and efficiency with history based prioritization âĂŞ an industrial case study. In *Proceedings of the 4th International Conference on Software Testing Verification and Validation*. IEEE Computer Society, 2011.

[7] E. Engström, P. Runeson, and M. Skoglund. A systematic review on regression test selection techniques. *Information and Software Technology*, 52(1):14–30, Jan. 2010.

[8] Y. Fazlalizadeh, A. Khalilian, M. A. Azgomi, and S. Parsa. Prioritizing test cases for resource constraint environments using historical test case performance data. In *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, pages 190 –195, Aug. 2009.

[9] M. Fewster and D. Graham. *Software Test Automation*. Addison-Wesley Professional, Sept. 1999.

[10] S. Mujtaba, R. Feldt, and K. Petersen. Waste and lead time reduction in a software product customization process with value stream maps. In *Software Engineering Conference (ASWEC), 2010 21st Australian*, pages 139–148. IEEE, 2010.

[11] R. Parasuraman, T. B. Sheridan, and C. D. Wickens. A model for types and levels of human interaction with automation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 30(3):286 –297, May 2000.

[12] K. Petersen and C. Wohlin. Context in industrial software engineering research. In *3rd International Symposium on Empirical Software Engineering and Measurement, 2009. ESEM 2009*, ESEM '09, pages 401–404, Washington, DC, USA, Oct. 2009. IEEE.

[13] A. Rainer and S. Beecham. A follow-up empirical evaluation of evidence based software engineering by undergraduate students. In *12th International Conference on Evaluation and Assessment in Software Engineering, University of Bari, Italy*, 2008.

[14] A. Rainer, T. Hall, and N. Baddoo. A preliminary empirical investigation of the use of evidence based software engineering by under-graduate students. https://uhra.herts.ac.uk/dspace/handle/2299/2270, 2006.

[15] G. Rothermel and M. J. Harrold. Analyzing regression test selection techniques. *IEEE Transactions on Software Engineering*, 22(8):529–551, Aug. 1996.

[16] G. Rothermel, R. H. Untch, C. Chengyun, and M. J. Harrold. Test case prioritization: an empirical study. In *Proceedings IEEE International Conference on Software Maintenance*, pages 179–188, 1999.

[17] T. B. Sheridan and W. L. Verplank. Human and computer control of undersea teleoperators. Technical report, July 1978.

[18] D. Talby, A. Keren, O. Hazzan, and Y. Dubinsky. Agile software testing in a large-scale project. *Software, IEEE*, 23(4):30–37, 2006.

[19] S. Yoo and M. Harman. Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability*, 22(2):67–120, Mar. 2012.