Licentiate Thesis Abstract: Evaluating Test Techniques using Fault and Failure Analysis

Sigrid Eldh
Ericsson AB & Mälardalens University
Stockholm & Västerås
sigrid.eldh@mdh.se

ABSTRACT
Testing is an important and costly activity in software industry today. This licentiate thesis is a product of four published papers, which are all case studies performed in an industrial setting. This work has led to the insight of how important – and difficult - preparations of controlled software experiments are. Our aim is to understand how to relate faults and their propagation to failures to different test techniques. We will give an overview of our four different papers, and some of our conclusions. The first paper presents a case study of testing in practice within the telecom domain. Next we performed an improvement in component test, by using known quality enhancing methods, and familiar test techniques, e.g. code coverage. This work resulted in a case study comparing the deployment in several design teams. With this as our basis, we created our position paper that describes a framework for evaluating test design techniques, and the process to do so. The final paper presents an attempt to conduct the first part of this process. We aimed to create a series of software components prepared with faults to serve as our controlled experiment, but the study resulted in an analysis of failures and corresponding faults in real industrial software. This failure-fault distribution is a partial result that clarified the difficulty in our endeavor. Based on this work, we have now outlined a series of problems that needs to be addressed before we will be able to continue creating our experiments and perform evaluations as originally planned.

Categories and Subject Descriptors
D.2.5 Testing and Debugging

General Terms
Experimentation, Verification.

Keywords
Fault, Failure, Test, Evaluation, Efficient test, Software

1. INTRODUCTION
Testing is an important area for research. The systems are becoming more complex and the amount of code is constantly increasing. The quality of the software is becoming more important. Industries are driven by business needs to minimize the cost of development and maintenance of systems. The majority of systems need to rely on its testing to demonstrate functional and non-functional aspects of the software. This licentiate thesis have has its base and is performed in an industrial setting. Even if an entire telecom node is available for our research, this system encompasses a code so large that to used it is causes difficulties in learning and adapting it.

The licentiate thesis is based on four published papers. Our first paper captures the industrial testing performed in everyday practice. Our second paper is post-case study, which compared quality and the application of the scheme called "Software Quality Rank" on more than 20 test design teams. This improvement program showed us the difficulty in deploying established research in reality. Then we found our mission – to focus on evaluation of suitable test design techniques for different levels, (e.g. component, integration and system level). By creating controlled experiments in an industrial setting, and performing a series of case studies on test techniques, we want to provide *guidelines* to industry. We are particularly interested in the following three aspects of test design techniques:

- Effectiveness – finding a variety of failures, amount
- Efficiency - how fast a failure can be found, i.e. in the entire process
- Applicability – how easy, straight-forward, and "useful" the technique is in an industrial setting.

Our third paper describes this approach, by defining a test framework and process. In this process, the first step is to create a controlled experiment. In our fourth paper, we performed a failure-fault analysis, to be able to re-inject "true" failures and their actual corresponding faults back into the code. During analysis, we encountered a series of difficulties, which have led us to refine our experiment set-up.

Most test techniques evaluations – and research on test techniques is based on software with very limited number of faults in fictive research settings. This makes it possible to draw conclusions only on that particular setting – or for that particular fault set. Drawing conclusions based on these limited results, poses problems for industry.

Therefore, we need a more substantial research in this area that makes it possible to utilize these results to advice on strategy for performing tests. Our current status and attention has through the work in this licentiate thesis culminated in an understanding of how to set up experiments. We hope to continue our work to improve software test technique evaluations.

2. RESEARCH IN AN INDUSTRIAL SETTING

We can confidently say we understand the need to simplify, minimize and abstract information from real systems to be able to perform research. The scale of industrial and commercial software systems surmounts the time and detail possible to encompass within one PhD frame. For our type of research, a long term focus is important to create an industrial guideline.

To perform experimental research on "real" software from industry creates a lot of difficulty. Introducing failures in commercial systems to study them is just not an option, and software needs to be separated to allow experiments. An alternative would be to use a small fraction of the software to experiment on. Unfortunately, "small" software do not execute sufficiently without its surrounding, and using simulators will not yield the same test results, which makes the results dubious. This meant that we needed to have some form of sub-system that can execute real industrial and commercial software. In telecom, these systems are often large and complex to use.

Our preparations of creating this controlled research experiment have led us to move an entire telecom node in a secluded environment to make us perform our experiments. It is difficult to use the node and also to utilizing it to its full potential. We have learned that we need extensive knowledge of the actual system and this makes the work even harder. A second important factor when performing research is the "observability" in the software. Observability is how easy it is to "see" the characteristics you are looking for. For us, this means if we introduce faults into the code, we want them to be "visible" in some form at execution of the software. We are experimenting further in this direction, trying to establish different type of systems "observability" factors. We want to see what makes a fault to propagate into a visible failure. We conclude that it is very easy to understand why most research in this area is using the prepared Space programs or the Siemens suite, which exists in a combined set with additions, in the system SIR [5].

3. TEST IN INDUSTRY, STATE OF PRACTICE

In our first paper we did an overview of our research subject that can be viewed as an initial case study. The title is "How to Save on Quality Assurance –Challenges in Software Testing" [1].

An important strategy for Industry is to collaborate with academia to find solutions to several difficult problems within software testing. In particular, this paper discusses test automation and component test. A lot of money can be saved by improving the test area, and this paper share some of the lessons learned to aid other businesses with the same endeavor. It can be viewed as explaining the setting of where the experiments are to be performed.

4. IMPROVING COMPONENT TEST

Our second paper titled "Experiments with Component Tests to Improve Software Quality" [2] presents an experiment comparing deployment of a special scheme for 23 different design teams in real industrial setting. The idea was that in commercial systems, time to market pressure often result in short-cuts in the design phase, where component test is most vulnerable. It is hard to define how much testing is cost effective by the individual developers, and hard to judge when testing is enough. Verification activities constitute a major part of the product cost. Failures discovered during later phases of product development escalate the cost substantially. To reduce cost in later stages of testing by reducing failures is important not only for Ericsson, but for any software producer. At Ericsson, we created a scheme, Software Quality Rank (SQR). SQR is a way to improve quality of components. SQR consists of five steps, where the first is where the actual "ranking" of components takes place. Then a selection of components is targeted for improvement in five levels. Most components are targeted for rank 3, which is the cost-efficient quality level. Rank 5 is the target for safety-critical code. The goal of SQR was to provide developers with a tool that prioritizes what to do before delivery to next system test phase. SQR defines a stepwise plan, which describes how much and what to test on component level for each rank. It gives the process for how to prioritize components; re-introduces reviews; requires usage of static analysis tools and defines what coverage to be achieved. The scheme has been used with great success at different design organizations within and outside Ericsson, and we believe it supports industry in defining what cost-efficient component test in a time-to market situation is.

5. A Framework for Test Evaluations

Our third paper "A Framework for Comparing Efficiency, Effectiveness and Applicability of Software Testing Techniques" [3], serves as the position paper for our entire research. It describes our aim to compare test techniques in an industrial setting. Although there is a multitude of test techniques, there are currently no scientifically based guidelines for the selection of appropriate techniques of different domains and contexts. For large complex systems, some techniques are more efficient in finding failures than others and some are easier to apply than others are.

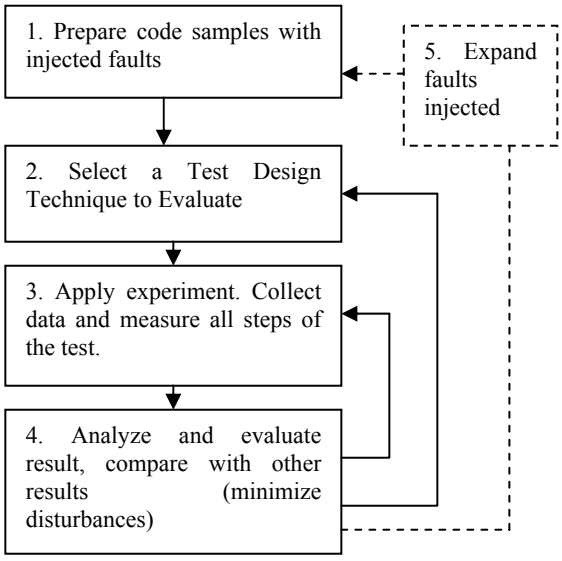

Figure 1. Overall process of evaluating test design techniques

From an industrial perspective, it is important to find the most effective and efficient test design technique that is possible to automate and apply. In this paper, we propose an experimental framework for comparison of test techniques with respect to

efficiency, effectiveness and applicability. We also plan to evaluate ease of automation, which has not been addressed by previous studies. We highlight some of the problems of evaluating or comparing test techniques in an objective manner. We describe our planned process for this multi-phase experimental study in Figure 1. This includes presentation of some of the important measurements to be collected with the dual goals of analyzing the properties of the test technique, as well as validating our experimental framework. Each of these process steps are described in more detail in the paper.

6. FAULT AND FAILURE ANALYSIS AND DISTRIBUTION

The work with creating a detailed process for evaluation was very valuable. We immediately started to tackle the first step in the process in figure 1. We had hoped there was existing fault classifications that could be injected. Instead, we found that the existing classification did not fulfill our purpose and had many limitations. We wanted to use more "intelligent" faults, and not use simple injection (mutation) with faults that could easily be found by the compiler. We want to create "high order semantic faults". We also wanted to use faults that would be visible at different levels of testing (see figure 2). Our hope was that we would find faults that would fulfill our needs. Since no existing list was available, we decided to look in our own system, and capture real faults that were found in testing, and see if we could re-inject them back.

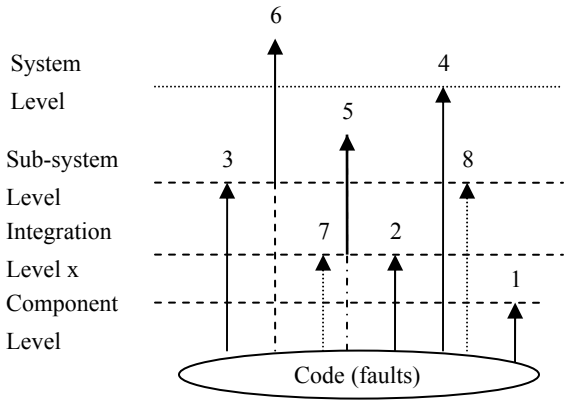

Figure 2. Fault propagation to failures, can be captured at different levels

This resulted in the fourth paper called "Component Testing is Not Enough - A Study of Software Faults in Telecom Middleware" [4]. The paper describes a high level classification, where we related failures to fault.

The interrelationship between software faults and failures is quite intricate and obtaining a meaningful characterization of it would definitely help the testing community in deciding on efficient and effective test strategies. Towards this objective, we have investigated and classified failures observed in a large complex telecommunication industry middleware system during 2003-2006. In this paper, we describe the process used in our study for tracking faults from failures along with the details of failure data. We present the distribution and frequency of the failures along with some interesting findings unravelled while analyzing the origins of these failures. Firstly, though "simple" faults happen, together they account for only less than 10%. The majority of faults come from either missing code or path, or superfluous code, which are all faults that manifest themselves for the first time at integration/system level; not at component level. These faults are more frequent in the early versions of the software, and could very well be attributed to the difficulties in comprehending and specifying the context (and adjacent code) and its dependencies well enough, in a large complex system with time to market pressures. This exposes the limitations of component testing in such complex systems and underlines the need for allocating more resources for higher level integration and system testing.

7. HYPOTHESIS AND PROBLEM AREAS

This section presents the hypothesis and problems areas of the thesis.

7.1 Hypothesis

It is possible to devise an efficient, effective and applicable test design technique to be applied at a specific level and specific phase, in a domain, given a failure-fault history. This is the main hypothesis for the overall PhD work and this main hypothesis can be divided in several problem areas, as explained below, where the first part is addressed within the licentiate thesis work.

7.2 Problem Areas

Through the main hypotheses we are trying to show that you can compare test techniques given some information, but it does not say how. Herein lays the problem. It is relatively straight forward to make results for one specific code in an instance of an application, but our aim is to try and make results more general and scalable. Below are the sub-hypotheses or "problem areas" we need to address, and answer. They describe what we need to provide to say that we have proved or disproved the hypothesis above. These additional problem areas and questions will be discussed in future work.

1. Fault-failure history analysis

To be able to draw conclusions from the fault-failure analysis, we need to know distribution and occurrence, but also to assure the main question, which is: What are the interesting faults to inject into software to base the experiment on and be able to draw conclusions from and makes it possible to generalize the result to other software? The work in [4], have given us insights in some of the questions that needs to be answered to be able to conclude this phase.

2. *Domain independence*

To be able to draw conclusions on if our result has domain independence, we need to understand what factors makes the results useful for other domains, i.e. generalized and scalable. We aim to approach this problem in a phased manner starting from a single system to multiple systems within a single domain, and then across domains. It should be possible to analyze software and system through its fault-failure distribution, and answering the above questions and draw conclusions on how that affects the result in testing. We need to understand which part of our results can be generalised and which cannot. Based on experience we can

see the relation of observability is different in different types of systems, as well as how easily some faults propagates into failures in some types of systems compared to others. At this point in time, we are not willing to draw any conclusions or make any proposals, since we have not yet started to explore this line of questions.

3. *Efficiency and Effectiveness*

We have suggested a set of measurements in [3] that would be possible to use to measure test technique efficiency and effectiveness. We have not yet had the possibility to test these measurements. But as discussed in future work, we have also different approaches on this.

4. *Applicability*

The area of applicability – or usability of a test techniques opens a wide set of questions that need to be answered to be able to continue our research. We believe measuring applicability is novel and will cast a light on test techniques in a new way. Not only have we in [3] identified a set of measurements to be used, but we also hope that dividing the test design and execution process will aid in focusing on the right problems.

8. RESEARCH RESULTS

This licentiate thesis has focused on establishing the industrial state of practice, and then defined a process to evaluate test techniques for efficiency, effectiveness and applicability. In principal, the contribution of creating such a process is not novel, but on a detailed level, applicability of test techniques is new. In particular how to create guidelines out of research, so results are both scalable and re-usable, is not an easy task – and the approach to applicability will contribute to better use of the techniques.

The status of this thesis is not so much in finding a lot of data and producing ready to use result, but to ask the right questions. The knowledge needed to be able to ask the right question is the main achievement, which challenges most previous test technique evaluations to the core. This gives a research result that opens up a series of areas for further studies. We are below summarizing the research result and what needs to be conducted to fulfil the goal of the area. We have also understood that the scope in this proposed PhD thesis might not fit the time-frame availability, and that we might need to take some short cuts to achieve partial goals within the given time.

9. CONCLUSIONS & FURTHER WORK

The described work has led us in a direction of truly aiming to connect a test technique to a failure (and its corresponding fault). The main difficulty is to define faults in a way that would transfer into many systems. Even if faults are not as unique as we thought, the high order semantics, the context and the fault constructs often contains a series of dependencies that behave or propagate differently in different systems. Therefore, better knowledge on faults, how they propagate to failures, observability related to the system, and ways to inject more complex “and real” semantic faults is our concern. We have planned a series of experiments that will run in parallel with other work. The second line of research is to better prepare and start evaluating the test design techniques. There is a need to find better description and definitions and start experimenting with test design techniques in different settings. We have already commenced work with different trials of evaluations.

10. ACKNOWLEDGMENTS

I would like to thank my supervisors Prof. Hans Hansson and Prof. Sasikumar Punnekkat. for their contribution to this licentiate thesis. The SAVE-IT Program at the Knowledge Foundation in cooperation with Mälardalens University and Ericsson AB funded this research.

11. REFERENCES

[1] Eldh, S.: How to Save on Quality Assurance – Challenges in Software Testing, Jornadas sobre Testeo de Software, p 103--121, ITI, Universidad Politecnica de Valencia, Valencia, Editor(s): Tanja E.J. Vos (2006)

[2] Eldh, S., Punnekkat, S., Hansson, H.: Experiments with Component Test to Improve Software Quality**,** *Proc. ISSRE*, IEEE Trollhättan, Sweden (2007)

[3] Eldh, S., Hansson, H., Punnekkat, S., Pettersson, A., Sundmark, D.: A Framework for Comparing Efficiency, Effectiveness and Applicability of Software Testing Techniques. *Proc. TAIC*, IEEE, London, UK (2006)

[4] Eldh, S., Punnekkat, S., Hansson, H., Jönsson., P.: Component Testing is Not Enough - A Study of Software Faults in Telecom Middleware, 19th IFIP International Conference on Testing of Communicating Systems TESTCOM/FATES, Springer LNCS, Tallinn, Estonia (2007)

[5] Hyunsook, D., Elbaum, S., Rothermel, G.: Infrastructure support for controlled experimentation with software testing and regression testing techniques, *Proc. Int. Symp. On Empirical Software Engineering*, ISESE '04, ACM Aug. (2004), 60 – 70