Exploratory Testing: A Multiple Case Study

Juha Itkonen and Kristian Rautiainen
Helsinki University of Technology, Software Business and Engineering Institute
P.O. BOX 9210, FIN-02015 Finland
firstname.lastname@hut.fi

Abstract

Exploratory testing (ET) – simultaneous learning, test design, and test execution – is an applied practice in industry but lacks research. We present the current knowledge of ET based on existing literature and interviews with seven practitioners in three companies. Our interview data shows that the main reasons for using ET in the companies were the difficulties in designing test cases for complicated functionality and the need for testing from the end user's viewpoint. The perceived benefits of ET include the versatility of testing and the ability to quickly form an overall picture of system quality. We found some support for the claimed high defect detection efficiency of ET. The biggest shortcoming of ET was managing test coverage. Further quantitative research on the efficiency and effectiveness of ET is needed. To help focus ET efforts and help control test coverage, we must study planning, controlling and tracking ET.

1. Introduction

Software developers and testers have always been doing exploratory testing. It is easy to understand this considering the short definition of exploratory testing: "Exploratory testing is simultaneous learning, test design, and test execution." [6]. This means that ET is testing without detailed pre-specified test cases, i.e., unscripted testing. ET is not a single testing technique or strategy; it is rather an approach to testing where test design is performed as part of test execution instead of having a test design phase before execution. The ET approach can be used for different types of testing using various techniques or methods. Many practitioners seem to promote ET as a valid approach to software testing and as a valuable part of an effective set of software quality assurance practices. ET is not a replacement for existing test-case based approaches, but a complementary testing approach suitable for certain situations which are not known due to lack of research. Many benefits of ET have been proposed, such as effectiveness, efficiency, the ability to utilize testers' creativity, and enabling rapid feedback [3, 4, 6, 7, 8, 11], but there is lack of scientific evidence of these benefits. Furthermore, very few explicit shortcomings of ET have been mentioned in the existing literature. This does not mean that there would not be any. Considering the potential benefits of ET and its seemingly wide use, we think that more research on the subject is merited.

The purpose of this paper is to introduce exploratory testing and study its use in three companies in order to increase the understanding of its applicability, benefits, and shortcomings. In this paper our intent is not to contrast ET and other testing approaches, but to describe the use of ET in industry. In addition, we bring forth several important research issues that should be studied to better understand the applicability and restrictions of ET as well as to develop new ways to better manage ET in order to lessen its shortcomings.

The rest of the paper is structured in the following way. In the next section we present the research methods used. Then we describe related work about exploratory testing, followed by the results of the case study. The paper is rounded up with discussion and implications for further work.

2. Research methods

In this section we present the research methods used in this study. Our research questions were: *1) What is the current knowledge of exploratory testing in literature? 2) How and why do companies use ET and with what results?* For the first research question we performed a literature study. For the second research question we selected a descriptive case study approach [13].

0-7803-9508-5/05/$20.00 (c)2005 IEEE

Table 1. Summary of the case companies

	Mercury	Neptune	Vulcan
Number of employees in software product development	15	30	40
Type of product	Professional application	Professional engineering design application	Professional engineering design application
Type of users	Professional control room workers	Professional engineers	Professional engineers
Training requirements of the product	Requires some training	Requires comprehensive training	Requires comprehensive training
Number of customers	< 10	> 100	> 1000
Number of users	< 100	> 1000	> 1000
Applied ET in its current form	Two months	Six months	Four years
Test approaches used	Only ET for the studied product	ET and automated scripted testing	ET for functional and smoke testing, test cases for system testing, and automated model-based testing
Number of interviewees	1	4	2
Formal testing training of the interviewees	None	None	None

We performed interviews and collected subjective evaluations of the benefits and shortcomings of ET and quantitative defect and effort data in three companies. In the following subsections we shortly describe the case companies and the data collection and analysis.

2.1. Case companies

The three case companies – Mercury, Neptune, and Vulcan – were selected based on accessibility through our research project in which they participate. Therefore we knew that exploratory testing was applied in these companies. The real names of the companies have been replaced with pseudonyms. Table 1 summarizes the characteristics of the companies. One of the companies is small with around 10 employees working with software development. The other two are bigger with around 30 and 40 employees in software development.

Neptune and Vulcan develop software application products for a large number of customers. Mercury has a more restricted customer segment. The products of all companies are systems for professional users. Using two of the systems requires comprehensive training. The products of Neptune and Vulcan have been on the market for several years, while the product of Mercury is new.

Mercury used only ET for their product. This was due to the immature development stage of the product. Neptune and Vulcan also used other testing approaches, although ET had a significant role in their testing efforts. In this paper we have focused on studying and describing how and why the companies used ET.

2.2. Data collection and analysis

We performed seven semi-structured thematic interviews lasting 40-70 minutes with persons that were performing exploratory testing in the case companies. At Vulcan, where ET had been purposefully used and improved for four years, we interviewed 2 persons. At Neptune, where a more systematic approach to ET had been introduced half a year ago, we interviewed 3 persons from the company and 1 outsourced tester that was a professional user of the system. The only interviewee at Mercury had used ET only a couple of times before the interview.

Each interviewee was interviewed separately using the same set of themes and questions. The questions were open ended and neutral, and the goal was to record the honest opinions and experiences without any leading of the interviewee. Two researchers were present at each interview. One researcher asked the questions and the other one made notes. The interviews were also recorded. The notes made during the interviews were used as a basis for the analysis. The notes were complemented and clarified by listening to the recordings. We used MindManager[1] to

[1] MindManager is a commercial software provided by the Mindjet Corporation. It is used for visualizing and managing information in the form of mind maps. For more information, see http://www.mindjet.com/

group the data and to create clusters that arose from the data.

We also collected quantitative data of ET effort as well as defect counts and types from the companies' defect and other tracking systems. We analyzed this data by deriving descriptive metrics to get a picture of the results of using ET. We compared these to the subjective evaluations of the interviewees for triangulation of the results.

3. Related work

In this section we describe the existing knowledge regarding exploratory testing. First, we describe how ET is defined in different sources. Then we summarize the claimed benefits and identified shortcomings of ET in literature. In the last subsection we describe the Session-Based Test Management approach for managing ET. Academic literature seems to lack research on exploratory testing. Therefore we must rely largely on text books, practitioner reports and electronic material available on the Internet.

3.1. Definition of exploratory testing

Exploratory testing is a loosely defined concept that was first introduced by Kaner et al. in the first edition of their book "Testing Computer Software" in 1988. They describe ET as a means to keep testing software after executing the scripted tests and avoid investing effort to carefully designing and documenting tests when the software is in an unstable state and could be redesigned soon. They also describe ET as a way of learning the system while designing the systematic test cases. However, they do not provide a precise definition of what is included in ET, how it should be performed, and what it is not. [8]

James Bach defines ET as "*...simultaneous learning, test design, and test execution*" [2]. Tinkham and Kaner give a slightly different definition: "*Any testing to the extent that the tester actively controls the design of the tests as those tests are performed and uses information gained while testing to design new and better tests*" [10]. According to Kaner, Bach, and Pettichord exploring means "*...purposeful wandering: navigating through a space with a general mission, but without a pre-scripted route. Exploration involves continuous learning and experimenting.*" [7]. They also propose that a tester should continuously learn about the product, its market, its risks and its previous defects in order to continuously build new tests that are more powerful than the older because they are based on the tester's continuously increasing knowledge.

Jonathan Bach stresses focusing on finding defects [3]. ET is a testing approach that is optimized to finding defects and puts less emphasis on test documentation. Finding defects is the primary purpose of ET and documenting the results of the testing is found more important than planning and scripting the test execution paths beforehand.

However, a certain degree of planning is needed for ET. Copeland [4] suggests performing "chartered exploratory testing" where the charter may define what to test, what documents are available, what tactics to use, what defect types to look for, and what risks are involved. The charter defines a mission for testing and works as a guideline.

The exploratory testing approach is recognized in the Software Engineering Body of Knowledge (SWEBOK): "*Exploratory testing is defined as simultaneous learning, test design, and test execution; that is, the tests are not defined in advance in an established test plan, but are dynamically designed, executed, and modified. The effectiveness of exploratory testing relies on the software engineer's knowledge, which can be derived from various sources...*" [6]. SWEBOK does not describe ET in more detail and does not provide any advice on how to apply ET or for which circumstances ET would be suitable.

Craig and Jaskiel emphasize the fact that in exploratory testing a tester can immediately expand testing into productive areas during testing [5]. They also state that in scripted testing a tester creates many tests that do not seem as useful during test execution as they seemed during test design, because they do not reveal defects.

From the above sources we can derive five properties that describe when testing is exploratory testing:

1) Tests are not defined in advance as detailed test scripts or test cases. Instead, exploratory testing is exploration with a general mission without specific step-by-step instructions on how to accomplish the mission.
2) Exploratory testing is guided by the results of previously performed tests and the gained knowledge from them. An exploratory tester uses any available information of the target of testing, for example a requirements document, a user's manual, or even a marketing brochure.
3) The focus in exploratory testing is on finding defects by exploration, instead of systematically producing a comprehensive set of test cases for later use.
4) Exploratory testing is simultaneous learning of the system under test, test design, and test execution.

5) The effectiveness of the testing relies on the tester's knowledge, skills, and experience.

In most of the sources ET is seen as a useful and effective approach to testing, but still, as a complementary approach to structured and systematic test-case based techniques.

3.2 Applicability of exploratory testing

James Bach describes contexts into which ET would fit well. First, ET fits situations where *rapid feedback or learning of the product is needed*. Second, ET fits situations where there is *not enough time for systematic testing approaches*. Third, ET is a good way to *investigate the status of particular risks*. Fourth, ET can be used to *provide more diversity* to scripted tests. Fifth, *regression testing based on defect reports* can be done by exploring. Sixth, ET fits well into *testing from an end-user viewpoint* based on, e.g., a user's manual. [2]

Copeland states that ET is valuable in situations where choosing the next test case to run cannot be determined in advance, but must be based on previous tests and results. ET can be used to explore the size, scope, and variations of a found defect to provide better feedback to developers. ET is also useful when test scripts become "tired", i.e., they are not detecting many defects anymore. [4]

Våga and Amland propose that ET should be planned as part of the testing approach in most of the software development projects [11]. They describe a case where ET was successfully used to test a web-based system in just two days, because that was all available time for testing.

3.3. Claimed benefits of exploratory testing

Exploratory testing has been advocated as a useful testing approach based on several benefits that it can provide for testing. In this section we summarize the benefits that are proposed in various sources [2, 7, 8].

The most commonly claimed benefit of ET is its ability to increase the effectiveness of testing in terms of number and importance of found defects. James Bach also suggests that in some situations ET can be orders of magnitude more efficient than scripted testing [2]. He supports his claim with some anecdotes from his personal experience. Exploratory testers can focus on the suspicious areas of the system based on the information of the actual behaviour of the system, instead of only relying on the specifications and design documents when planning the tests.

A second benefit of exploratory testing is the simultaneous learning. When testers are not following pre-specified scripts, they are actively learning about the system under test and gaining knowledge about the behavior and the failures in the system. This is claimed to help testers come up with better and more powerful tests as testing proceeds.

A third benefit is the ability to minimize preparation documentation before executing testing. This is an advantage in a situation where the requirements and the design of the system change rapidly or at the early stage of product development when some parts of the system are implemented, but the probability for major changes is still high.

A fourth benefit is the ability to perform exploratory testing without comprehensive requirements or specification documentation, because exploratory testers can easily utilize all the experience and knowledge of the product gained from various other sources.

A fifth benefit is the rapid flow of feedback from testing to both developers and testers. This feedback loop is especially fast, because exploratory testers can react quickly to changes to the product and provide test results back to developers.

3.4. Identified shortcomings of exploratory testing

We had a hard time finding explicit references to shortcomings of ET in the existing literature. One identified shortcoming of exploratory testing is the difficulty of tracking the progress of individual testers and the testing work as a whole. It is considered hard to find out how the work proceeds, e.g., the feature coverage of testing, because there is no planned low-level structure that could be used for tracking the progress. [3, 11]

Lee Copeland [4] points out that ET has no ability to prevent defects. Designing the test cases in scripted testing can begin during the requirements gathering and design phases and thus reveal defects early.

3.5 Session-based test management

Exploratory testing should not be unplanned, unstructured, or careless testing without any strategy or goals [5]. Exploratory testing can be as disciplined as any other intellectual activity [2]. It can be structured, managed and planned as long as the planning is not extended to describe detailed tests on the level of test cases and execution steps.

Jonathan Bach has published an approach to ET called Session-Based Test Management (SBTM) [3]. The session-based approach brings a clear structure to loosely defined exploratory testing. It is an approach to planning, managing, and controlling exploratory testing in short (almost) fixed-length sessions. Each of these sessions is planned in advance on a charter sheet. The charter is a high level plan for a test session. It does not pre-specify the detailed test cases executed in each session. A clearly defined report is produced and metrics are collected during the session. The results are debriefed afterwards between the tester and the test manager.

Experiences of using a similar approach have been presented by Lyndsay and van Eeden [9]. The approach of Lyndsay and van Eeden also includes methods for controlling the scope of the testing, assessing and tracking the coverage of the tests, and assessing risks and prioritizing the tests. The session-based approaches seem to provide tools for managing ET and alleviation to the planning as well as progress and coverage tracking problems of ET.

4. Results

In this section we present the results from our case study. First we present the reported reasons for using ET in the case companies. Then we present the different ways of performing ET in the companies. The following subsections describe the perceived benefits and shortcomings of ET. In the last subsection we present the collected measurement data.

4.1. Reasons for using exploratory testing

The companies reported several reasons for using ET. In Table 2 we have summarized all reasons mentioned in more than one company. All companies agreed that writing test cases for everything is difficult and laborious. At Neptune and Vulcan, when using the system, a task could be performed in so many ways that it was impossible to write test cases for all possible combinations. Therefore an exploratory approach was regarded a natural choice. At Mercury, the interviewee stated that it could have taken up to a week to write a list of test cases and still some important test cases might have been overlooked or forgotten. Instead, the interviewee preferred using the same time to testing and giving feedback to development.

Another reason that all companies agreed upon was using ET as a way of testing the software from a user's viewpoint. When performing ET, they tested larger combinations of functionality together from the viewpoint of performing the tasks of a real professional user of the system. Exploring real use scenarios helped find peculiarities and usability problems in the software. At Neptune, the user manual was used to structure exploratory testing from the user's viewpoint. This also enabled validating the correctness and usefulness of the manual. Testing from the user's viewpoint was also reported challenging, since it requires very strong domain knowledge from the tester.

Mercury and Neptune mentioned ET as a natural way of doing testing, since it emphasizes utilizing the testers' experience and creativity to find defects during test execution. At Vulcan, this was mentioned when talking about using ET to regression testing. When new features have been developed or defects have been corrected, ET is useful for testing that nothing else has been broken. The interviewees at Vulcan commented that even if test cases are used as a basis for testing, ET allows the tester to look at the tested feature(s) as a whole. This makes it easier to spot problems that might go unnoticed if the tester was only following a script. Also, the exploratory attitude helps the tester to follow hunches and thus find defects, for example, in unexpected combinations of features. In this way ET was by the interviewees regarded both as an independent way of testing and as a complementary testing approach to using predefined test cases.

Table 2. Reported reasons for using ET in the case companies

Reasons for using ET	Mercury	Neptune	Vulcan
The software can be used in so many ways or there are so many combinations between different features that writing detailed test cases for everything is difficult, laborious, and even impossible.	X	X	X
It suits well to testing from a user's viewpoint.	X	X	X
It emphasizes utilizing the testers' experience and creativity to find defects.	X	X	X
It helps provide quick feedback on new features from testers to developers.	X	X	X
It adapts well to situations, where the requirements and the tested features change often, and the specifications are vague or incomplete.	X	X	
It is a way of learning about the system, the results of which can be utilized in other tasks, such as customer support and training.		X	X

All companies had found ET practical for giving fast feedback to developers regarding newly developed features. When the developer(s) had completed a feature, a tester would quickly explore the new feature and give feedback to the developer(s). The feedback could range from reporting defects to pointing out usability problems or misconceptions regarding the customer requirements. This information could then be used to steer the development, if necessary.

At Mercury and Neptune, the requirements and developed features could change often during development. Therefore it was considered a waste of time to write detailed test cases during the early phases of development. Also, the specifications were deliberately left incomplete or even vague when development started. In these conditions, ET was considered a logical approach to testing.

At Neptune and Vulcan, ET was viewed as a way to learn about the system under test. This was considered valuable, because the information and experience could then be used to support the tester's other tasks, since there were no full-time testers in any of the companies. For example, the information was used in planning additions to the set of existing automated tests. Also, some of the testers worked in customer support and training. The information they got from doing ET helped them better understand the questions received from customers and prepare the answers. The information and experience was also useful for preparing training material.

4.2. Ways of utilizing exploratory testing

Based on our interviews we identified six different ways of utilizing exploratory testing in the three case companies: session-based ET, functional testing, exploratory smoke testing, exploratory regression testing, subcontracted ET, and freestyle ET. These were applied in different phases of the development life cycle and are explained in more detail in the subsections below. Even if the companies had many different approaches to exploratory testing at the process level, the actual testing work of each individual tester did not seem to differ a lot. All interviewees described their testing as an intuitive and ad-hoc process of trying to find defects or verify changes. The interviewees had different goals for the testing, but none of the interviewees could describe any intentional test strategies or techniques that they used when exploring the software. When asked about trying out different combinations and finding equivalence classes or boundary values, four of the seven interviewees told that they try to find out combinations and boundaries to intentionally break the system. However, none of the interviewees claimed they do this systematically.

4.2.1. Session-based exploratory testing. At Neptune and Mercury, a session-based exploratory testing approach was used. In this approach, testing was organized in short (0,5-3 hours) test sessions during which the tester accomplished one planned testing task (for example, "test the insert note functionality using the different views of the system") and tried to concentrate and focus on that task without any interruptions or other disturbance. Mercury's sessions were shorter (59 minutes on average) than Neptune's sessions (113 minutes on average). Most of the interviewed persons found it beneficial to isolate the testing time into focused sessions without other tasks or interruptions. The sessions were planned using short descriptions that were called charters as in [3].

The charters described briefly the testing task, goals of the test session, and the target of testing, i.e., the product and the feature to be tested. At Mercury, the tester's testing actions on a high level of abstraction were logged at the end of the test session charter during the test session. At Mercury, the person who performed the testing wrote the charter and at Neptune, the charters were written by the developer whose responsibility the tested functionality was. There was no systematic higher level planning or control of how the test sessions were allocated, and no tracking of which parts of the system were covered by the test sessions and how thoroughly.

The main results that were reported from the sessions were found defects in the form of entries in the defect tracking system. At Mercury, the interviewee recorded the found defects directly into the defect tracking system during the session, but at Neptune the interviewees wrote down the found defects briefly by hand or into a text editor during the session and reported the defects into the defect tracking system only after the session. The interviewees felt that this way of working was better because it helped them keep their concentration on the work at hand.

4.2.2. Functional testing of individual features. Vulcan used ET for testing individual features right after the feature was implemented. This was performed by individuals of the requirements management team and focused on testing whether the implementation corresponded to the requirements and the designer's actual ideas of the specified functionality or not. This enabled fast feedback to the developers in the early phase of the development life cycle.

4.2.3. Exploratory smoke testing. Vulcan used an exploratory smoke-testing approach after the implementation phase, when a new revision of the software was released to testing, which could occur once a week. Each of these releases was smoke tested by the service team. This exploratory testing took from half an hour to a day and was guided by a "heading-level" list of the areas to be tested. The tester went through the list with the intent of identifying defects in or changes to the existing functionality and formulating a quick overall picture of the general quality of the release. In addition, the tester checked every fix and enhancement that was implemented in the release to ensure that the reported fixes actually had been properly performed and worked as the service team member would expect from the end-user point of view.

4.2.4. Exploratory regression testing. Both Neptune and Vulcan used exploratory testing to verify fixes and changes after implementing a single fix. This testing was driven by announcements from developers that some defect was fixed or enhancement implemented. The tester held a short testing session to verify the fix, typically without any planning or formal tracking or control. The result of this session was informally communicated to the developer or, if it was a defect fix, the defect was just checked as closed into the defect tracking system.

This approach differs from the typical regression testing described in textbooks. In these companies regression testing was not performed exhaustively over the whole system. Rather it concentrated on the changes and fixes made and, based on the tester's experience, exploring possible new and related defects caused by the fixes. The main reasons mentioned for this kind of "limited" regression testing were lack of time or resources for complete regression testing of the system.

4.2.5. Subcontracted exploratory testing. As a special form of testing, Neptune used real users of the system as subcontracted testers. They hired two experienced professional users of their system to test their upcoming release. This testing was organized by features and the task of the testers was to explore each feature of the software to accomplish real working scenarios.

4.2.6. Freestyle exploratory testing. At Vulcan, many parts of the organization performed unmanaged exploratory testing as part of their other duties. The common way of doing it was testing the latest alpha or beta release. For example, at customer services this was a part of the everyday work.

An interviewee at Vulcan also mentioned that they quite often use exploratory testing as part of systematic system testing to explore functionality beyond the documented test cases with the intent of finding more defects and defects that are not straightforward to find.

4.3. Perceived benefits of exploratory testing

Some of the perceived benefits of ET were mentioned in Section 4.1 as part of the reasons for using ET. In this section we present additional benefits the interviewees mentioned.

The testing performed using ET is more versatile and goes deeper into the tested feature(s) according to all interviewees. Five out of the seven interviewees mentioned that they tend to test things that they might not have included in a test plan or among test cases. Examples of such tests include testing the dependencies of new and existing features based on expertise and knowledge of the system. This would normally not be included as test cases because they are written based on the requirements for the new features. Another example of versatility is retesting a corrected defect. Interviewees at both Neptune and Vulcan mentioned that they do not just retest in the same way as before, but explore for possible new defects at the same time. At Neptune, this was realized in some cases so that the person that found and reported the defect was not the one who tested the correction. In this way the other tester, who explored the software from a different viewpoint, could use his experience to try to find new and related defects.

The effectiveness and efficiency of ET was perceived high by the interviewees, although with some reservations. At Vulcan, both interviewees concluded that ET helped them find important defects in a short amount of time, but if a less experienced person with less domain knowledge would do the testing, the results might not be so good. They also claimed that more defects were found in system testing using ET than using test-case-based testing and they implied this could be because the test cases are designed to verify that the system works and that the testers use ET with a more destructive attitude. At Neptune, one of the interviewees thought that ET is efficient in the short term considering used hours and the number of found defects. However, in the long run it is difficult to determine the efficiency of ET, because it is hard to estimate the coverage of the tests and many things may go untested and thus unnoticed causing problems in the future. Another interviewee at Neptune thought ET was effective, but he also mentioned that

using ET to test features of a complex system is very time consuming.

Getting an overall picture of the quality of the system quickly is one aspect of efficiency that was mentioned by interviewees at Neptune and Vulcan. At Neptune, this was considered important because the information and overall picture gained from ET was used as a basis for prioritizing the work towards the end of the project. At Vulcan, when the testers get a new version of the system they can quickly determine the quality level based on their prior experience. They can also quickly verify that the visual look and feel of the system is consistent with the historical look and feel, which is considered important since the system has been in use for many years and existing customers would probably not welcome inconsistent changes.

4.4. Perceived shortcomings of exploratory testing

Coverage in one form or another was the biggest shortcoming of ET mentioned by all the interviewees. At Mercury, the biggest challenge concerning coverage was planning and selecting what to test with ET. The interviewee admitted that everything could not be tested with ET, because it is time consuming and there are not enough testers. The factor of limited time and testers was also mentioned at Vulcan, where the interviewees admitted that not everything could be tested. It was a question of prioritizing testing to potential weak spots in the system under test and trying to allocate time of domain experts for testing. This, combined with scarce documentation of the testing itself, created another challenge, namely following up what had been tested and what had not. At Neptune, coverage was considered a challenge from two viewpoints: 1) when testing new features, some things most certainly were left untested, especially concerning resulting effects in the old features, and 2) a tester can never guess all the ways in which the customer might use the system.

Being able to utilize the testers experience and creativity was mentioned as a reason and benefit for using ET in all the companies. However, this was also mentioned as a shortcoming. In both Neptune and Vulcan the interviewees agreed that relying on the expertise of the testers made ET more prone to human errors than systematic testing. At Neptune, one interviewee commented that it was impossible to find testers with enough experience to act as professional users. Another comment from both Neptune and Vulcan was that all testers have different backgrounds and experience and thus perform ET from different viewpoints. Again, this was seen both as a strength and weakness, especially regarding the versatility of testing this implied.

The repeatability of defects was seen as a shortcoming of ET at Neptune. This was attributed to the complex system that permitted many ways of performing tasks, and each task could require up to a hundred or more steps. When a defect was spotted, it could take hours to repeat it, so that it could be properly reported in the defect database. However, the macro recording capability of the system could be used to alleviate the problems. Repeatability problems had also occurred at Vulcan, but they had mainly happened due to memory leaks and could probably not have been repeated even if a detailed script had been followed. At Mercury, repeatability was not considered a problem, because of the extensive session logs written during the ET sessions.

4.5. Defect and effort data

We collected quantitative defect and effort data from the defect and other tracking systems of the case companies. The collected data is summarized in Table 3. It helps in describing how the companies used ET. The available data has limitations which makes it hard to make conclusions based on it. At Vulcan, a session-based testing approach was not used which means that there is no session-specific data available.

Table 3. Summary of defect and effort data

	Neptune	Mercury	Vulcan	
			Functional	Smoke
Total number of found defects	169	34	103 / release	31 / release
Total effort (hours)	36	4	160 / release	NA
Number of sessions	17	4	NA	NA
Average session length (minutes)	113	59	NA	NA
Average defects / session	9.9	8.5	NA	NA
Average defects / hour	4.8	8.7	0.6	NA
Serious defects	NA	15 %	NA	NA

The session data of Neptune and Mercury is collected from the test sessions that they have performed so far. Vulcan's data is collected from four successive product releases and presented as averages. Two different ways of doing exploratory testing at Vulcan could be taken into account in data collection: functional testing (see Section 4.2.2) and smoke testing (see Section 4.2.3). There was no effort data of smoke-testing available. The number of serious defects could not be determined for Neptune and Vulcan.

Within each case we cannot say whether the numbers in Table 3 show good or bad performance since we have no comparative data of other testing approaches in the companies. Based on a cross-case comparison, however, defect detection efficiency (Average defects / hour in Table 3) is much higher where a session-based approach has been used. This may partly be because the tester can stay focused by avoiding interruptions during sessions. The interviewees at Vulcan admitted that they encountered many interruptions during testing, but they did not see it as a problem. The defect rate per hour metric of Vulcan's functional testing might also be biased. We cannot ensure that all detected defects have been logged into the defect tracking system or that effort has been correctly and exactly allocated to ET.

The difference between the defect rates per hour at Neptune and Mercury may be explained by the maturity of the products. Mercury's product is new and it is logical that it might contain more defects to be detected.

5. Discussion and further work

The purpose of this paper was to study what the current knowledge of exploratory testing is in literature, how and why companies use ET, and with what results. We presented the current knowledge of ET, its claimed benefits, applicability, and shortcomings based on the existing literature, which we found very scarce. We conducted a case study in three software product companies. We looked at the reasons for using ET, how ET is applied in the companies, and what the perceived benefits and shortcomings of ET are. The resulting description of the use of ET in the companies is the main contribution of this paper. The results of our study support many of the claimed benefits of existing literature and reveal some new findings. Next we discuss the limitations of our study, our findings, and propose ideas for further research (ideas marked in *italics*).

We recognize that there are many limitations to our study, especially concerning the small number of case companies. There were only three case companies, which makes generalization of the results difficult. Neptune and Vulcan are alike in many respects, while Mercury is smaller and has a little bit different customer segment, but it still means that the representativeness of the case companies is limited. Also, all case companies are in the product business, so we cannot say if or how ET would work in, e.g., bespoke software projects. Another limitation is that we focused on the use of ET in the companies and not the whole testing approach and the role ET plays in it. In view of the limitations of the study, our findings should be considered more suggestive than conclusive. *A more comprehensive case study should be made, with a bigger and more versatile sample of companies*. However, our findings add to the body of knowledge concerning exploratory testing and provide some food for thought.

The data did not fully support the claimed benefit of increased productivity. Many interviewees felt that ET is an effective way of finding defects, especially defects that were hard to find. Still, they also considered it time consuming to explore complicated functionality carefully. The defect and effort data in Table 3 seems to support claims about defect detection efficiency, at least concerning session-based ET. In studies we can use as comparison, Anderson et al. [1] found an average defect detection efficiency of less than 3 defects per hour for usage-based testing, where the tester tests a piece of software based on test cases derived from use cases. Wood et al. found an average defect detection efficiency of 2.47 defects per hour for functional testing [12]. The defect detection efficiency for Neptune and Mercury was 4.8 and 8.7 defects per hour respectively (see Table 3). Also, 15% of the found defects at Mercury were serious, which gives some support for the effectiveness of session-based ET. However, these issues *require studies where a reliable comparison of efficiency and effectiveness can be made between ET and e.g. test-case based testing*.

Both the literature and the interviewees agreed that one of the main reasons for using ET is being able to utilize the testers' creativity and experience during test execution, instead of spending much time on designing test cases prior to test execution. The interviewees stressed the difficulty of creating good test cases because the systems were so complicated and provided so many options for the user. However, the interviewees stated clearly their concern for the strong reliance on the expertise of the individual tester in ET. It is hard to find testers with enough domain expertise to act as professional users and different testers may focus on different things when testing. This makes evaluating the quality of testing challenging. These

shortcomings have not been explicitly addressed in the existing literature. An interesting question for further research is *what importance domain knowledge and testing skills play in finding defects*. None of the interviewees had received any formal testing training. *Would the interviewees have found more defects if they had received testing training? Would a professional tester be able to find relevant defects without domain knowledge?* These questions remain open for further research.

Our study did not reveal any intentional test techniques or strategies used for exploring. This could be due to lack of testing training. We feel that *further research on ET techniques and strategies is needed*, because they might increase the effectiveness of ET.

One new finding of our study that has not been mentioned in existing literature was the use of ET for learning the system for other purposes than better testing and finding defects more effectively. In two of the three case companies one of the reasons for using ET was learning the features and behavior of the system, e.g. to help prepare training material and answers in customer service.

Based on our study the biggest shortcoming of ET is coverage. Only selected parts of the system can be tested because of time pressure, but in the case companies there were no established ways of planning and prioritizing what to use ET for. This, combined with insufficient mechanisms for following up test progress, resulted in unknown coverage, which was a concern for the interviewees. It seems that we need *more research to find reliable techniques for planning, controlling and tracking ET to help focus ET efforts and help control coverage*. The session-based approach suggested in [3, 9] seems promising in this respect.

It seems that exploratory testing is an accepted approach to testing in industry and in the future we will continue our research efforts on ET. As ET seems to work well as a complementary testing method, *more detailed case studies are needed focusing on the role of ET in a comprehensive testing process*. We plan to conduct more case studies to gain better understanding of the role, and benefits of exploratory testing in software development. We plan to focus our efforts on session-based approaches and hope to find ways of coping with the shortcomings and challenges of ET while utilizing the full potential of the benefits it can provide. Another area of interest is exploration techniques. We plan to arrange student and industrial experiments to get results of the effect of using different techniques in exploratory testing.

References

[1] C. Andersson, T. Thelin, P. Runeson and N. Dzamashvili, "An Experimental Evaluation of Inspection and Testing for Detection of Desing Faults", in *Proceedings of International Symposium on Empirical Software Engineering,* 2003, pp. 174-184.

[2] J. Bach, "Exploratory Testing", in *The Testing Practitioner,* Second ed., E. van Veenendaal Ed., Den Bosch: UTN Publishers, 2004, pp. 253-265.

[3] J. Bach, "Session-Based Test Management", *STQE,* vol. 2, no. 6, 2000.

[4] L. Copeland, *A Practitioner's Guide to Software Test Design,* Boston: Artech House Publishers, 2004.

[5] R.D. Craig and S.P. Jaskiel, *Systematic Software Testing,* Boston: Artech House Publishers, 2002.

[6] IEEE, "Guide to the Software Engineering Body of Knowledge", IEEE., Tech. Rep. IEEE - 2004 Version, 2004.

[7] C. Kaner, J. Bach and B. Pettichord, *Lessons Learned in Software Testing,* New York: John Wiley & Sons, Inc., 2002.

[8] C. Kaner, J. Falk and H.Q. Nguyen, *Testing Computer Software,* New York: John Wiley & Sons, Inc., 1999.

[9] J. Lyndsay and N. van Eeden, "Adventures in Session-Based Testing", 2003, Accessed 2005 04/25, http://www.workroom-productions.com/papers/AiSBTv1.2.pdf

[10] A. Tinkham and C. Kaner, "Exploring Exploratory Testing", 2003, Accessed 2005 04/25, http://kaner.com/pdfs/ExploringExploratoryTesting.pdf

[11] J. Våga and S. Amland, "Managing High-Speed Web Testing", in *Software Quality and Software Testing in Internet Times,* D. Meyerhoff, B. Laibarra, van der Pouw Kraan,Rob and A. Wallet Eds., Berlin: Springer-Verlag, 2002, pp. 23-30.

[12] M. Wood, M. Roper, A. Brooks and J. Miller, "Comparing and Combining Software Defect Detection Techniques: A Replicated Empirical Study", *ACM SIGSOFT Software Engineering Notes,* vol. 22, no. 6, 1997, pp. 262-277.

[13] R.K. Yin, *Case Study Research: Design and Methods,* London: SAGE Publications, 1994.