Exploratory Testing of Large-Scale Systems – Testing in the Continuous Integration and Delivery Pipeline

Torvald Mårtensson[1](✉), Daniel Ståhl[2], and Jan Bosch[3]

[1] Saab AB, Linköping, Sweden
torvald.martensson@saabgroup.com
[2] Ericsson AB, Linköping, Sweden
daniel.stahl@ericsson.com
[3] Chalmers University of Technology, Gothenburg, Sweden
jan@janbosch.com

Abstract. In this paper, we show how exploratory testing plays a role as part of a continuous integration and delivery pipeline for large-scale and complex software products. We propose a test method that incorporates exploratory testing as an activity in the continuous integration and delivery pipeline, and is based on elements from other testing techniques such as scenario-based testing, testing in teams and testing in time-boxed sessions. The test method has been validated during ten months by 28 individuals (21 engineers and 7 flight test pilots) in a case study where the system under test is a fighter aircraft. Quantitative data from the case study company shows that the exploratory test teams produced more problem reports than other test teams. The interview results show that both engineers and test pilots were generally positive or very positive when they described their experiences from the case study, and consider the test method to be an efficient way of testing the system in the case study.

Keywords: Continuous delivery · Continuous integration · Exploratory testing · Large-scale systems · Software testing

1 Introduction

Exploratory testing was coined as a term by Cem Kaner in the book "Testing Computer Software" [1] 1988, and was then expanded upon as a teachable discipline by Kaner, Bach and Pettichord in their book "Lessons Learned in Software Testing" [2] in 2001. The test technique combines test design with test execution, and focuses on learning about the system under test.

Different setups exist for planning, execution and reporting exploratory testing. Testing can be organized as charters [3, 4] or tours [3, 5] which are conducted as sessions [3, 4] or threads [3]. Janet Gregory and Lisa Crispin [3] describe the test technique with the following words: "Exploratory testers do not enter into a test session with predefined, expected results. Instead, they compare the behavior of the system against what they might expect, based on experience, heuristics, and perhaps oracles. The difference is subtle, but meaningful." The core of the test technique is the focus on learning, shown in for example Elisabeth Hendricksson's [4] definition of exploratory testing: "Simultaneously designing and executing tests to learn about the system, using your insights from the last experiment to inform the next".

Coevally with the evolution of exploratory testing, continuous integration and other continuous practices emerged during the 1990s and early 2000s. The exact moment for the birth of each practice is up for debate. Continuous integration is often referred to as a term coming from either Kent Beck's book "Extreme Programming" [6] in 1999 or Martin Fowler's popular article [7] in 2006, and the term continuous delivery seems to have been established by Jez Humble and David Farley in the book "Continuous Delivery" [8] in 2011. Automated testing is described as a corner stone of continuous practices, and automated tests tend to be the focus when test activities are assembled to a continuous integration and delivery pipeline (shown in Fig. 1). This pipeline splits the test process into multiple stages, and is described with different terminology by Duvall as "stage builds" [9], by Larman and Vodde as "multi-stage CI system" [10] or by Humble and Farley as the "deployment pipeline" or "integration pipeline" [8]. Humble and Farley [8] include exploratory testing in the final stage before release to the customer. We believe that exploratory testing also can play an important role early in the integration flow, especially when developing large-scale systems with many dependencies between the subsystems.

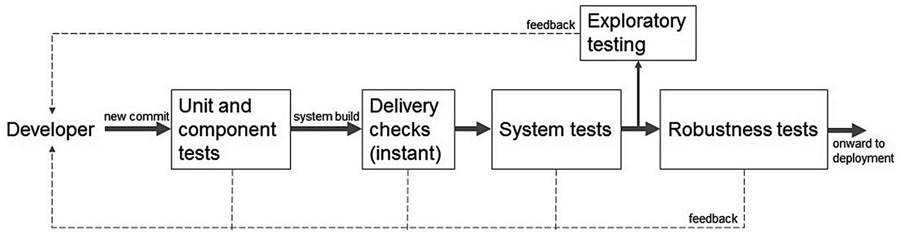

Fig. 1. An example of a continuous integration and delivery pipeline (including exploratory testing), showing the flow of test activities that follows a commit of new software

Based on this, the topic of this paper is to answer the following research question: *How can exploratory testing be used in the continuous integration and delivery pipeline during development of large-scale and complex software products?*

The contribution of this paper is three-fold. First, it presents a test method for large-scale and complex software products. Second, the paper shows how exploratory testing plays a role as part of a continuous integration and delivery pipeline for large-scale and complex software products. Third, it provides quantitative data and interview results from a large-scale industry project. The remainder of this paper is organized as follows. In the next section, we present the research method. This is followed in Sect. 3 by a study of related literature. In Sect. 4 we present the test method, followed by validation in Sect. 5. Threats to validity are discussed in Sect. 6. The paper is then concluded in Sect. 7.

2 Research Method

The first step to answer the research question stated in Sect. 1 was to conduct a systematic literature review (according to Kitchenham [11]), which is presented in Sect. 3. The question driving the review was "Which test methods related to exploratory testing and testing of large-scale and complex systems have been proposed in literature?"

The test method for exploratory testing of large-scale systems was developed based on related published literature and experiences in the case study company. The test method was validated using the following methods to achieve method and data triangulation [12]:

- *Systematic literature review*: Comparison of the test method and related work found in literature.
- *Validation interviews*: Interviews with 18 engineers and 7 flight test pilots who used the test method during ten months.
- *Analysis of quantitative data*: Exploratory analysis of quantitative data (problem reports and time used in the test rig) retrieved from the case study.

Interviews were held with 25 of the 28 individuals who were participating in the test activity in the case study. The remaining three had in two cases changed jobs, and was in one case on parental leave. The interviews were conducted as semi-structured interviews, held face-to-face or by phone using an interview guide with pre-defined specific questions. The interview questions were sent to the interviewee at least one day in advance to give the interviewee time to reflect before the interview. The questions in the interview guide were:

- How would you describe your experiences from [name of the test activity in the project]?
- What did you like or not like about…
  - The planning meetings?
  - The briefings before testing?
  - The test sessions in the rig?
  - The debriefings after testing?
- What do you like or not like about [name of the test activity in the project] compared to other types of test activities?
- Are you interested in participating in this type of activity again?

The interview results were analyzed based on thematic coding analysis as described by Robson [13, pp. 467–481], resulting in three main themes corresponding to the characteristics of the test method (each supported by statements or comments by between 15 and 20 of the interviewees). The process was conducted iteratively to increase the quality of the analysis. Special attention was paid to outliers (interviewee comments that do not fit into the overall pattern) according to the guidelines from Robson [13], in order to strengthen the explanations and isolate the mechanisms involved.

Detailed data on e.g. types of scenarios selected by the test teams, types of issues found during the test sessions or detailed interview results are not included in this research paper due to non-disclosure agreements with the case study company.

3 Reviewing Literature

3.1 Criteria for the Literature Review

To investigate whether solutions related to the research question have been presented in published literature, a systematic literature review [11] was conducted. A review protocol was created, containing the question driving the review ("Which test methods related to exploratory testing and testing of large-scale and complex systems have been proposed in literature?") and the inclusion and exclusion criteria. The inclusion criterion and the exclusion criterion for the review are shown in Table 1.

Table 1. Inclusion and exclusion criteria for the literature review

Inclusion criterion	Yield
Publications matching the Scopus search string `TITLE-ABS-KEY ("exploratory testing" AND software)` on March 27, 2017	52
Exclusion criterion	**Remaining**
Excluding duplicates, conference proceedings summaries and publications with no available full-text	39

To identify published literature, a Scopus search was conducted. The search was updated before writing this research paper, in order to include the state-of-the-art. The decision to use only one indexing service was based on the fact that we in previous work have found Scopus to cover a large majority of published literature in the field, with other search engines only providing very small result sets not already covered by Scopus.

3.2 Results from the Literature Review

An overview of the publications found in the systematic literature review is presented in Table 2. The review of the 39 publications retrieved from the search revealed that five of the publications were not directly related to exploratory testing. These papers use the term "exploratory testing" as a keyword without a single mention in the article itself or only mentioning it in passing. In addition to that, one of the papers was a poster which contained the same information as another paper found in the search.

Ten of the papers were related to methods and tools, typically combining two test techniques such as model-based testing and exploratory testing [14–18]. Two papers proposed different approaches to combine script-based testing and exploratory testing [19, 20] and one paper described how to extract unit tests and from exploratory testing [21]. One paper discussed "guidance for exploratory testing through problem frames" [22] and finally one paper investigated the feasibility of using a multilayer perceptron neural network as an exploratory test oracle [23].

Table 2. An overview of the publications found in the systematic literature review

Topic of the publications	Number of papers
Not relevant	5
Poster	1
Methods/tools	10
Effectiveness and efficiency of test methods	14
How exploratory testing is used	5
Reporting experiences	4
Summary	39

Fourteen of the publications discussed the effectiveness and efficiency of different test methods. Two of those were systematic literature reviews [24, 25] and one combined a systematic literature review and a survey [26]. Eight papers [27–34] compared exploratory testing and scripted testing (also referred to as test case based testing or confirmatory testing). The comparisons were based on either true experiments or experiences from industry projects. Sviridova et al. [35] discuss effectiveness of exploratory testing and proposes to use scenarios. Micallef et al. [36] discuss how exploratory testing strategies are utilized by trained and not trained testers, and how this affect the type of defects the testers find. Raappana et al. [37] report the effectiveness of a test method called "team exploratory testing", which is defined as a way to perform session-based exploratory testing in teams.

Five papers describe in different ways how exploratory testing is used by the testers, based on either a true experiment [38], a survey [39], video recordings [40] or interviews [41, 42]. Itkonen and Rautiainen [42], Shoaib et al. [38] and Itkonen et al. [40] describe how the tester's knowledge, experiences and personality are important while performing exploratory software testing in industrial settings. Itkonen et al. [41] present the results of a qualitative observation study on the manual testing practices, and presents a number of exploratory strategies: "User interface exploring", "Exploring weak areas", "Aspect oriented testing", "Top-down functional exploring", "Simulating a real usage scenario", and "Smoke testing by intuition and experience".

Finally, four papers [43–46] report experiences from exploratory testing in industry, but without presenting any quantitative or qualitative data as validation. Suranto [44] describes experiences from using exploratory testing in an agile project. Pichler and Ramler [46] describes experiences from developing and testing a visual graphical user interface editor, and touches upon the use of exploratory testing as part of an iterative development process. Gouveia [43] reports experiences from using exploratory testing of web applications in parallel with automated test activities in the continuous integration and delivery pipeline.

In summary, we found no publications that discussed exploratory testing in the context of large-scale and complex software system. Some publications touched on topics related to the subject, such as iterative development and continuous integration (which are commonly used during development of large-scale and complex software systems).

4 Exploratory Testing of Large-Scale Systems

4.1 Characteristics of the Test Method

The *test method for exploratory testing of large-scale systems* is based on related published literature and experiences from the case study company. In this case, exploratory testing is used to test a large-scale and complex system, which may consist of a range of subsystems that are tightly coupled with a lot of dependencies.

The motivation behind developing the test method was an interest in the case study company to increase test efficiency, and to find problems related to the integration of subsystems earlier in the development process. The transformation to continuous development practices implies a transformation from manual to automated testing. This requires large investments, both a large initial investment in implementing automated test cases and later costs for maintaining the test cases to keep up with changes in the system under test. For test activities that is likely to not remain static (the same specification is run over and over again) it is an alternative to utilize the flexibility of experienced engineers in manual test activities.

The test method is designed to complement automated testing in the continuous integration and delivery pipeline, and to provide different feedback and insights than the results from an automated test case. The characteristics of the test method are:

- *Exploratory testing as an activity in the continuous integration and delivery pipeline*: Testing is conducted with an exploratory approach where the testers simultaneously learn about the system's characteristics and behavior. Testing is done regularly on the latest system build, which has passed the test activity in the preceding step in the continuous integration and delivery pipeline.
- *Session-based testing in teams with experienced engineers representing different subsystems:* Testing is conducted in time-boxed sessions by teams of hand-picked experienced engineers, representing the different subsystems of the product. If the size or complexity of the system under test cannot be covered by a single team, the test scope can be split between several teams.
- *Scenario-based testing with an end-user representative as part of the test team*: Testing is conducted in scenarios, which represent how the product will be used by the end-user. An end-user representative is participating in both planning and test execution, securing that the scenarios are reflecting appropriate conditions.

The characteristics of the test method are in different ways described or touched upon in published literature. Exploratory testing has been described (at least briefly) in the context of agile or iterative development [3, 44, 46] and one report describes how exploratory testing is used in the "continuous integration pipeline" [43]. Exploratory testing is often combined with the use of sessions [3, 4, 28, 37, 40] and the concept of testing in teams has been described [37] or at least touched upon [3]. There are also publications that enhance the importance of experience and knowledge [38, 40, 42]. The use of scenarios is also described in different ways [3, 5, 35, 41], but not specifically with an end-user representative as part of the test team.

4.2 Using the Test Method

The test team work together in planning workshops, test sessions and debriefing meetings (shown in Fig. 2).

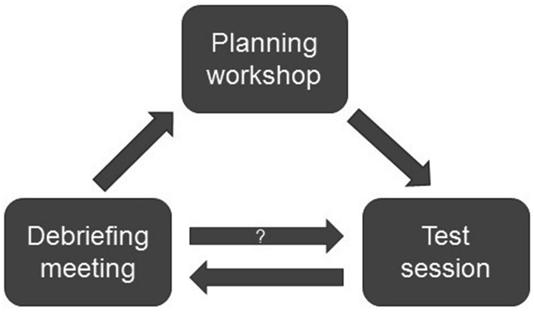

Fig. 2. The flow between planning meetings, test sessions and debriefing meetings

At the *planning meeting*, the test team discusses ideas for testing that could result in finding uncovered problem areas. The team members prioritize and group the test ideas into scenarios, which could be executed during a test session. A scenario is a chain of events that could be introduced by either the product's end-user, derive from a problem in the product's software or hardware systems, or be coming from other systems or the environment where the product is operated (e.g. change of weather if the product is a car). The test team is monitoring the reports from other test activities in the continuous integration and delivery pipeline, in order to follow new or updated functions or new problems that have been found which could affect the testing.

During the *test session*, the scenarios are tested in a test environment which is as production-like as possible. The test environment must also be equipped so that the test team is able to test fault injection and collect data using recording tools. Before the test session the team must also decide on test approaches for the planned test sessions: Should the team observe as many deviations as possible or stop and try to find root causes? Should the team focus on the intended scope or change the scope if other issues come up?

The *debriefing meeting* is used by the team to summarize the test session. The responsibility to write problem reports or follow up open issues found in the test session is distributed among the team members. The team should consider if a problem should have been caught at a test activity earlier in the pipeline, and report this in an appropriate way. Decisions are made if the tested scenarios should be revisited at the next session or not. The team should also discuss how team collaboration and other aspects of test efficiency could be improved.

5 Validation

5.1 The Case Study

The case study company is developing airborne systems and their support systems. The main product is the Gripen fighter aircraft, which has been developed in several variants. Gripen was taken into operational service in 1996. An updated version of the aircraft (Gripen C/D) is currently operated by the air forces in Czech Republic, Hungary, South Africa, Sweden and Thailand. The next major upgrade (Gripen E/F) will include both major changes in hardware systems (sensors, fuel system, landing gear etc.) and a completely new software architecture.

The test method described in Sect. 4 was applied to a project within the case study company for ten months. The system under test was the aircraft system with functionality for the first Gripen E test aircraft, which was tested in a test rig. The test pilot was maneuvering the aircraft in a cockpit replica, which included real displays, panels, throttle and maneuvering stick. In the rig the software was executing on the same type of computers as in the real aircraft. The aircraft computers were connected to an advanced simulation computer, which simulated the hardware systems in the aircraft (e.g. engine, fuel system, landing gear) as well as a tactical environment. A visual environment was presented on an arc-shaped screen. The test team communicated with the pilot from a test leader station in a separate room. From the test leader station the tester could observe the pilot's displays and the presentation of the aircraft's visual environment. The test team could also observe the behavior of the software in the aircraft computers and inject faults in the simulator during flight (e.g. malfunction of a subsystem in the aircraft).

Continuous integration practices such as automated testing, private builds and integration build servers were applied in the development of software for the Gripen computer systems. When a developer committed new software to the mainline, the new system baseline was tested in multiple stages in a pipeline similar to the example shown in Fig. 1. All test activities on unit, component and system level which were effectuated up to weekly frequency were automated tests, followed by exploratory testing and other manually executed test activities.

Testing was conducted in sessions, starting with four hours per session which after two months was changed to three hours. The testing started with two teams, followed by a third team after a month. The teams tested at a frequency of one test session per week for two weeks out of three, meaning that generally two of the three teams tested every week. The testers were handpicked from the development teams, all being senior engineers representing different subsystems in the aircraft. A test pilot (from the flight test organization) was maneuvering the aircraft in the simulator. The engineers (in total 21 individuals) were allocated to the three test teams, each of which focused on one cluster of subsystems in the aircraft. The last two months the teams were merged to one test team, due to that no new functions were introduced and not so many new problems where found during the test sessions.

5.2 Validation Interviews

The interviewed 18 engineers who participated in the test activity were generally very experienced, all with many years of experience from industry software development. The interviewed 7 pilots were all employed as flight test pilots, with training from military pilot schools and experience from many years of service in both the air force and as test pilots in the industry. Both engineers and test pilots were generally positive or very positive when they described their experiences. "Relevant and good testing", to quote one of the test pilots. One of the engineers described it with the following words: "It was fantastic! We identified a lot of problems. And we learned how the system worked."

The three test teams used the way of working described in Sect. 4 with planning meetings, test sessions and debriefing meetings. The interviewees described that they "built a backlog" of things to test at the planning meetings, which was then used during the upcoming test sessions. The planning meetings were described with words as "creative" or "at least as interesting as the testing itself". Interviewees from one of the test teams described that they at first did very little preparations before the testing, resulting in some unprepared and inefficient test sessions. This changed when the team focused more on the planning meetings.

All teams held a short briefing (10–15 min) right before the test session, in order to go through the program for the test session. This was appreciated by both engineers and test pilots, as it gave everyone a picture of what would happen. During the briefing roles and responsibilities were also clearly distributed (communicating with the pilot, taking notes etc.). The testing itself was generally described as efficient, where engineers and the test pilot were working together as a team. One voice asked for better tools for some of the fault injection procedures, and someone else asked for better recording capabilities. After the test session the team had a short debriefing, with the purpose to summarize the findings and decide who was to write problem reports or further examine open issues. The teams often also had a follow-up meeting the day after the test, focusing on improving test efficiency and ways of working.

Both the engineers and the test pilots were generally very generous with comments and thoughts regarding their experiences from the test activities. Many engineers described their experiences with a lot of enthusiasm, and in some cases even referring to the testing as "great fun". The experiences shared by the interviewees are summarized in themes corresponding to the characteristics of the test method:

- Exploratory testing as an activity in the continuous integration and delivery pipeline
- Session-based testing in teams with experienced engineers representing different subsystems
- Scenario-based testing with an end-user representative as part of the test team

Exploratory testing as an activity in the continuous integration and delivery pipeline: Both engineers and test pilots described the benefits with exploratory testing, where the test teams not plainly follow the instructions in a test case step by step. As one interviewee described it: "We could test according to the ideas we had. We wanted to understand the system that we were building and to find the weaknesses in the system". A few interviewees also described that they during this test activity were looking for the root cause of the problems that were found, whereas they in other test activities just wrote down a brief description of the problem. Besides talking about the benefits from the higher level of freedom, many engineers also described the need for structure and discipline. A field of improvement seemed to be communication of the results from other test activities in the continuous integration and delivery pipeline. Several interviewees described situations where the team was not sure if a problem was already known, or even if a function was complete or still under development. However, according to the interviewees the synchronization with other test activities improved over time.

Session-based testing in teams with experienced engineers representing different subsystems: Almost all engineers described benefits from testing in teams. According to the interviewees, many of the questions that came up at a test session could be solved directly during the test session. Another engineer described that "the quality of the problem reports improves if there are people from different subsystems participating at the test". The engineers described that they were "learning about the system" and "learning about other subsystems". A few voices talked about the importance of having the right people onboard, referring to personality as well as knowledge and experience from the different subsystems of the product. To have a team of six or up to eight people participating during the same test session could also be challenging. Several interviewees described that it sometimes was difficult to see what was going on at the displays, and it was important that the test leader was good at involving all team members in the test process.

Scenario-based testing with an end-user representative as part of the test team: Almost all interviewees described or touched upon that scenarios was a good way to test the complete system. Both engineers and test pilots described that most of the other test activities focused on a subsystem in the aircraft, whereas this test activity focused on the complete aircraft. The interviewees seemed to like to use scenarios as a description of the tests, seeing it as a description that everyone could understand and more flexible than a traditional test case. Several engineer commented on the value to use a real test pilot, who could describe how the product would be used by the end user. The test pilots also described that they could "learn a lot from the engineers". To quote one of the test pilots: "During this test activity the engineers who design the product came in direct contact with the pilots who use it". A few voices (especially from the test pilots) asked for more clear objectives with each scenario test.

One of the questions in the interview guide asked the interviewee to compare the exploratory test activity and other types of test activities. None of the interviewees wanted to describe one way of testing as better than the other, but did instead in different ways describe that exploratory testing and specification-based testing are different types of testing with different purposes. To quote one of the engineers: "Testing according to [a test specification] verifies that the function is implemented according to the specification. This type of testing checks that it is good enough, that we can use the product."

Two of the engineers were a bit less positive than the others. One of them described it like "it never worked quite well", but explained this with that the subsystem he was representing had very little coupling to other subsystems. The other engineer described his situation in the following way: "I was never fully in, I do not know why. I had no clear vision of the whole system. I wished I had known more about my own subsystem, to be able to answer questions from the others."

The last question in the interview guide was if the interviewee was interested in participating in this type of activity again. Twenty-three of the 25 interviewees answered the question with "yes". Some of the engineers and some of the test pilots added that their participation were depending on priority decisions from management. Two of the participants answered the question with "maybe". One of them just added "we'll see when the question comes up". The other described himself as "not completely negative, but not completely positive either" but did not expand this further.

5.3 Problem Reports and Testing Time

Each test session in the case study resulted in a number of found defects in the system or open issues. The open issues were discussed with other developers or system managers, which sometimes clarified that the behavior was according to design, and sometimes confirmed that this was a defect in the system. All defects were documented as problem reports in the organization's issue management tool. All test sessions were conducted in one of the test rigs. The test rig was a scarce and valued test resource, as it was constructed with the same bespoke hardware as a real aircraft and a complex system for the visual environment (as described in Sect. 5.1).

Figure 3 shows for every month during the test period how many percent of all problem reports that month that came from the exploratory test teams. The figure also shows how many percent of all time in the rig that month (rig maintenance not included) that were booked by the exploratory test teams. The figure shows that except for May (and July when almost no testing was done due to vacation period) the exploratory test teams produced a larger share of the problem reports than the exploratory test teams' share of the rig time. As problem reports from other test activities were also written based on testing in other rigs and test environments, the share of time in all related rigs and test environments is actually even lower.

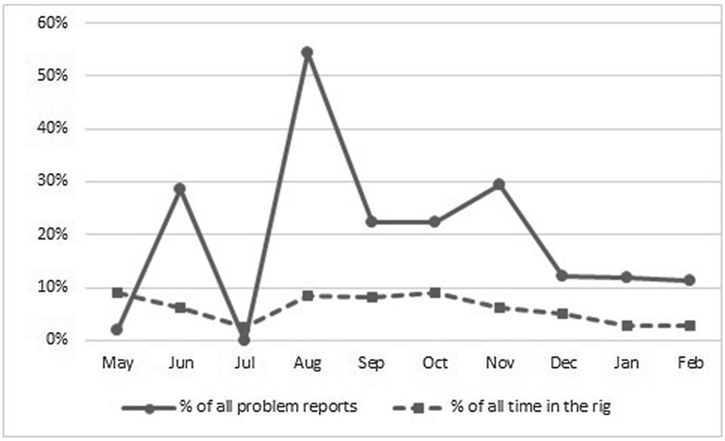

Fig. 3. The exploratory test teams' share of problem reports (in percent) and share (in percent) of all time in the rig

Figure 4 shows how the problem reports from the exploratory test teams are distributed over the ten months when the test activity was conducted. The figure also shows how all testing time in the rig used by the exploratory test teams is distributed over the same period of time. Figures 3 and 4 together show that the three test teams started a bit slow, and did not generate so many problem reports the first month. This changed during June, and peaked during August. Then the trends seem to stabilize for three months, followed by a period of time when the activity was run less intensively due to that no new functions were introduced.

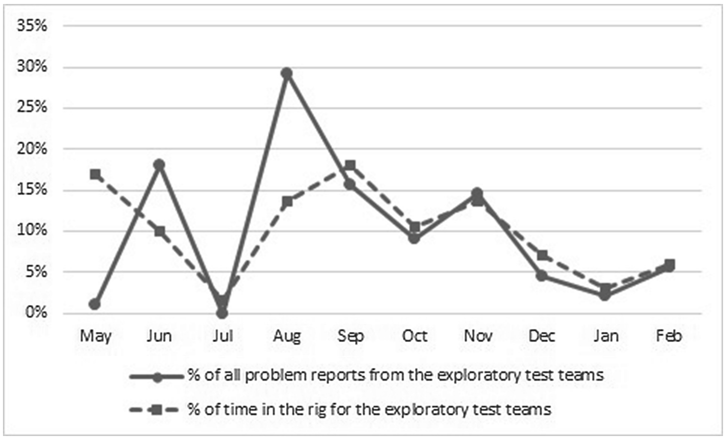

Fig. 4. Distribution of problem reports and testing time for the exploratory test teams

6 Threats to Validity

6.1 Threats to Construct Validity

One must always consider that a different set of questions and a different context for the interviews can lead to a different focus in the interviewees' responses. In order to handle threats against construct validity, the interview guide was designed with open questions (presented in Sect. 2). In this paper, we present both the interview guide and the background for both the interviewees and the case study in order to provide as much information as possible about the context.

The test rig was considered to be a scarce and valued resource. Therefore, we measure the number of problem reports (defects found) per unit of rig time in order to discuss the efficiency of the test method. We do not claim to discuss efficiency on more general terms, such as comparing the importance of the problem reports from different types of test activities (which we consider much harder to measure or quantify).

The observed effectiveness of exploratory testing in terms of number of problem reports may have been influenced by a focus on new functionality. It is conceivable that using the test method with a more clear focus on regression testing would provide a different result.

It is conceivable that the effectiveness of the studied test method is affected by the knowhow and experience of the participants in the study. As the studied test method was new for the participants, the study represents an early usage phase or basically the introduction of the test method. Results and feedback from participants may be different once the test method has turned into an established practice.

6.2 Threats to Internal Validity

Of the 12 threats to internal validity listed by Cook, Campbell and Day [47], we consider Selection, Ambiguity about causal direction and Compensatory rivalry relevant to this work:

- *Selection*: Interviews were held with 25 of the 28 individuals who were participating in the test activity. The remaining three had in two cases changed jobs, and was in one case on parental leave. As the interview series managed to cover all of the participants that were present at the company, there was no selection of interviewees.
- *Ambiguity about causal direction*: While we in this study discuss correlation, we are very careful about making statements regarding causation. Statements that include cause and effect are collected from the interview results, and not introduced in the interpretation of the data. Due to this, we consider this threat to be mitigated.
- *Compensatory rivalry*: When performing interviews and comparing scores or performance, the threat of compensatory rivalry must always be considered. The questions in our interviews were deliberately designed to be value neutral for the participants, and not judging performance or skills of the interviewee or the interviewee's organization. Generally, the questions were also designed to be opened-ended to avoid any type of bias and ensure answers that were open and accurate. However, our experiences from previous work is that we found the interviewees more prone to self-criticism than to self-praise.

6.3 Threats to External Validity

The validation of the test method is based on interviews and quantitative data from a single company. It is conceivable that the findings from this study are only valid for this company, for companies that operate in the same industry segment (military aircraft), or for similar products in different types of industry segments (e.g. other types of vehicles). The characteristics of the test method are in different ways described in related work (as described in Sect. 4), which we argue supports the generalizability of the results of this study (external validity). We have also presented detailed information about both the case study company and the project in the case study, in order to support attempts to replicate our results in other studies.

7 Conclusion

In this paper, we have discussed how exploratory testing can be used in the continuous integration and delivery pipeline during development of large-scale and complex software products. We have proposed a new test method with the following characteristics:

- Exploratory testing as an activity in the continuous integration and delivery pipeline
- Session-based testing in teams with experienced engineers representing different subsystems
- Scenario-based testing with an end-user representative as part of the test team

The characteristics of the test method are in different ways described or touched upon in published literature, which we argue strengthens the validation of the test method. However, none of the found publications presents a test method focusing on large-scale and complex systems, which we argue strengthens this paper's position as a valid contribution. The test method has been validated in a case study, where the system under test was a fighter aircraft. The test method was used during ten months by 28 individuals (21 engineers and 7 flight test pilots). Validation is based on quantitative data and interviews with 25 of the 28 participants.

Quantitative data from the case study company (presented in Sect. 5.3) shows that the exploratory test teams produced more problem reports than other test teams. The three test teams started a bit slow, and did not generate so many problem reports the first month. This changed the following months, and the number of problem reports peaked during the fourth month.

The interview results (summarized in Sect. 5.2) show that the characteristics of the test method are considered valuable by the interviewed 18 engineers and 7 flight test pilots, and that they consider the test method to be an efficient way of testing the system in the case study. Both engineers and test pilots embraced exploratory testing, and appreciated more freedom. Coordination with other test activities in the continuous integration and delivery pipeline was described as a problem at the beginning of the case study, but this improved later on. Many of the engineers described that they were able to test that the subsystems worked together, and that they learned about other subsystems due to that the team consisted of engineers from different subsystems. Engineers and test pilots thought that testing with scenarios was a good way to test the complete system, and described it as valuable to have the test pilot as an end-user representative participating in the test activity. The interviewees were generally positive or very positive when they described their experiences from the case study, using phrases like “relevant and good testing” or “we learned a lot”.

Consequently, we find that the test method presented in this paper succeeds in incorporating exploratory testing in the continuous integration and delivery pipeline and is an efficient test method for large-scale and complex software products. This is a significant result, as we see great value in how automated testing and exploratory testing could be complementing one another, each mitigating the weaknesses of the other by addressing unique concerns. Whereas automated test activities in the pipeline are able to rapidly provide feedback to developers and to verify requirements, exploratory testing can provide more in-depth insights about the system under test. Based on this research study, we believe that exploratory testing should be used in a continuous integration and delivery pipeline, preferably to test new functions and systems in a large-scale system.

7.1 Further Work

As the validation in this paper is based on a single case study, this calls for validation from other case studies using the same test method. As a suggestion, the system under test could be another type of vehicle, such as a car or a truck. This type of study could also be combined with the use of other methods to compare the efficiency of the test method (preferably using quantitative data).

References

1. Kaner, C.: Testing Computer Software. TAB Books, Blue Ridge Summit (1988)
2. Kaner, C., Bach, J., Pettichord, B.: Lessons Learned in Software Testing. Wiley, New York (2001)
3. Gregory, J., Crispin, L.: More Agile Testing. Addison Wesley, Boston (2015)
4. Hendrickson, E.: Explore It! The Pragmatic Bookshelf, Dallas (2013)
5. Whittaker, J.: Exploratory Software Testing. Addison Wesley, Boston (2010)
6. Beck, K.: Extreme Programming Explained: Embrace Change, 1st edn. Addison-Wesley Professional, Boston (1999)
7. Fowler, M.: Continuous integration (2006). http://www.martinfowler.com/articles/continuousIntegration.html
8. Humble, J., Farley, D.: Continuous Delivery. Addison Wesley, Boston (2011)
9. Duvall, P.: Continuous Integration. Addison Wesley, Boston (2007)
10. Larman, C., Vodde, B.: Practices for Scaling Lean & Agile Development – Large, Multisite, and Offshore Product Development with Large-Scale Scrum. Addison Wesley, Boston (2010)
11. Kitchenham, B.: Procedures for performing systematic reviews. Keele UK Keele University, vol. 33, pp. 1–26 (2004)
12. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Softw. Eng. **14**(2), 131–164 (2009). doi:10.1007/s10664-008-9102-8
13. Robson, C., McCartan, K.: Real World Research, 4th edn. Wiley, London (2016)
14. Frajtak, K., Bures, M., Jelinek, I.: Exploratory testing supported by automated reengineering of model of the system under test. Cluster Comput. **20**(1), 855–865 (2017). doi:10.1007/s10586-017-0773-z
15. Frajtak, K., Bures, M., Jelinek, I.: Model-based testing and exploratory testing: is synergy possible? In: 6th International Conference on IT Convergence and Security, ICITCS 2016 (2016). 7740354
16. Gebizli, C.Ş., Sözer, H.: Automated refinement of models for model-based testing using exploratory testing. Softw. Qual. J., 1–27 (2016). doi:10.1007/s11219-016-9338-2
17. Schaefer, C.J., Do, H.: Model-based exploratory testing: a controlled experiment. In: Proceedings of IEEE 7th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2014, pp. 284–293 (2014). 6825674

18. Schaefer, C., Do, H., Slator, B.M.: Crushinator: a framework towards game-independent testing. In: Proceedings of 2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013, pp. 726–729 (2013). 6693143
19. Shah, S.M.A., Gencel, C., Alvi, U.S., Petersen, K.: Towards a hybrid testing process unifying exploratory testing and scripted testing. J. Softw. Evol. Process. **26**(2), 220–250 (2014). doi:10.1002/smr.1621
20. Rashmi, N., Suma, V.: Defect detection efficiency of the combined approach. In: Satapathy, S., Avadhani, P., Udgata, S., Lakshminarayana, S. (eds.) ICT and Critical Infrastructure. AISC, vol. 249, pp. 485–490. Springer, Cham (2014). doi:10.1007/978-3-319-03095-1_51
21. Kuhn, A.: On extracting unit tests from interactive live programming sessions. In: Proceedings of International Conference on Software Engineering, pp. 1241–1244 (2013). 6606688
22. Kumar, S., Wallace, C.: Guidance for exploratory testing through problem frames. In: Proceedings of Software Engineering Education Conference, pp. 284–288 (2013). 6595262
23. Makondo, W., et al.: Exploratory test oracle using multi-layer perceptron neural network. In: 2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016, pp. 1166–1171 (2016). 7732202
24. Thangiah, M., Basri, S.: A preliminary analysis of various testing techniques in Agile development - a systematic literature review. In: Proceedings of 3rd International Conference on Computer and Information Sciences, ICCOINS 2016, pp. 600–605 (2016). 7783283
25. Garousi, V., Mäntylä, M.V.: A systematic literature review of literature reviews in software testing. Inf. Softw. Technol. **80**, 1339–1351 (2016)
26. Ghazi, A.N., Petersen, K., Börstler, J.: Heterogeneous systems testing techniques: an exploratory survey. In: Winkler, D., Biffl, S., Bergsmann, J. (eds.) SWQD 2015. LNBIP, vol. 200, pp. 67–85. Springer, Cham (2015). doi:10.1007/978-3-319-13251-8_5
27. Itkonen, J., Mantyla, M.V., Lassenius, C.: Test better by exploring: harnessing human skills and knowledge. IEEE Softw. **33**(4), 90–96 (2016). 7155417
28. Afzal, W., et al.: An experiment on the effectiveness and efficiency of exploratory testing. Empirical Softw. Eng. **20**(3), 844–878 (2015). doi:10.1007/s10664-014-9301-4
29. Itkonen, J., Mäntylä, M.V.: Are test cases needed? Replicated comparison between exploratory and test-case-based software testing. Empirical Softw. Eng. **19**(2), 303–342 (2014). doi:10.1007/s10664-013-9266-8
30. Shah, S.M.A., et al.: Exploratory testing as a source of technical debt. IT Prof. **16**(3), 44–51 (2014). 6475929
31. Shah, S.M.A., Alvi, U.S., Gencel, C., Petersen, K.: Comparing a hybrid testing process with scripted and exploratory testing: an experimental study with practitioners. In: Cantone, G., Marchesi, M. (eds.) XP 2014. LNBIP, vol. 179, pp. 187–202. Springer, Cham (2014). doi:10.1007/978-3-319-06862-6_13
32. Prakash, V., Gopalakrishnan, S.: Testing efficiency exploited: scripted versus exploratory testing. In: 2011 3rd International Conference on Electronics Computer Technology, ICECT 2011, vol. 3, pp. 168–172 (2011). 5941824
33. Itkonen, J., Mäntylä, M.V., Lassenius, C.: Defect detection efficiency: test case based vs. exploratory testing. In: Proceedings of 1st International Symposium on Empirical Software Engineering and Measurement, ESEM 2007, pp. 61–70 (2007). 4343733
34. Do Nascimento, L.H.O., Machado, P.D.L.: An experimental evaluation of approaches to feature testing in the mobile phone applications domain. In: Workshop on Domain-Specific Approaches to Software Test Automation - In Conjunction with the 6th ESEC/FSE Joint Meeting, DoSTA 2007, pp. 27–33 (2007)

35. Sviridova, T., Stakhova, D., Marikutsa, U.: Exploratory testing: management solution. In: 2013 12th International Conference on the Experience of Designing and Application of CAD Systems in Microelectronics, CADSM 2013, p. 361 (2013). 6543293
36. Micallef, M., Porter, C., Borg, A.: Do exploratory testers need formal training? An investigation using HCI techniques. In: Proceedings of 2016 IEEE International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2016, pp. 305–314 (2016). 7528977
37. Raappana, P., et al.: The effect of team exploratory testing - experience report from F-Secure. In: Proceedings of 2016 on IEEE International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2016, pp. 295–304 (2016). 7528976
38. Shoaib, L., Nadeem, A., Akbar, A.: An empirical evaluation of the influence of human personality on exploratory software testing. In: 2009 IEEE 13th International Multitopic Conference, INMIC 2009 (2009). 5383088
39. Pfahl, D., et al.: How is exploratory testing used? A state-of-the-practice survey. In: International Symposium on Empirical Software Engineering and Measurement (2014)
40. Itkonen, J., Mäntylä, M.V., Lassenius, C.: The role of the tester's knowledge in exploratory software testing. IEEE Trans. Softw. Eng. **39**(5), 707–724 (2013). 6298893
41. Itkonen, J., Mäntylä, M.V., Lassenius, C.: How do testers do it? An exploratory study on manual testing practices. In: 2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009, pp. 494–497 (2009). 5314240
42. Itkonen, J., Rautiainen, K., Exploratory testing: a multiple case study. In: 2005 International Symposium on Empirical Software Engineering, ISESE 2005, pp. 84–93 (2005). 1541817
43. Gouveia, N.: Agile testing on an online betting application. In: Sharp, H., Hall, T. (eds.) XP 2016. LNBIP, vol. 251, pp. 193–200. Springer, Cham (2016). doi:10.1007/978-3-319-33515-5_16
44. Suranto, B.: Exploratory software testing in agile project. In: 2015 2nd International Conference on Computer, Communications, and Control Technology, Art Proceeding, I4CT 2015, pp. 280–283 (2015). 7219581
45. Moss, C.: Big visible testing. In: Proceedings of AGILE 2013, pp. 94–100 (2013). 6612884
46. Pichler, J., Ramler, R.: How to test the intangible properties of graphical user interfaces? In: Proceedings of the 1st International Conference on Software Testing, Verification and Validation, ICST 2008, pp. 494–497 (2008). 4539578
47. Cook, T.D., Campbell, D.T., Day, A.: Quasi-Experimentation: Design & Analysis Issues for Field Settings, vol. 351. Houghton Mifflin, Boston (1979)