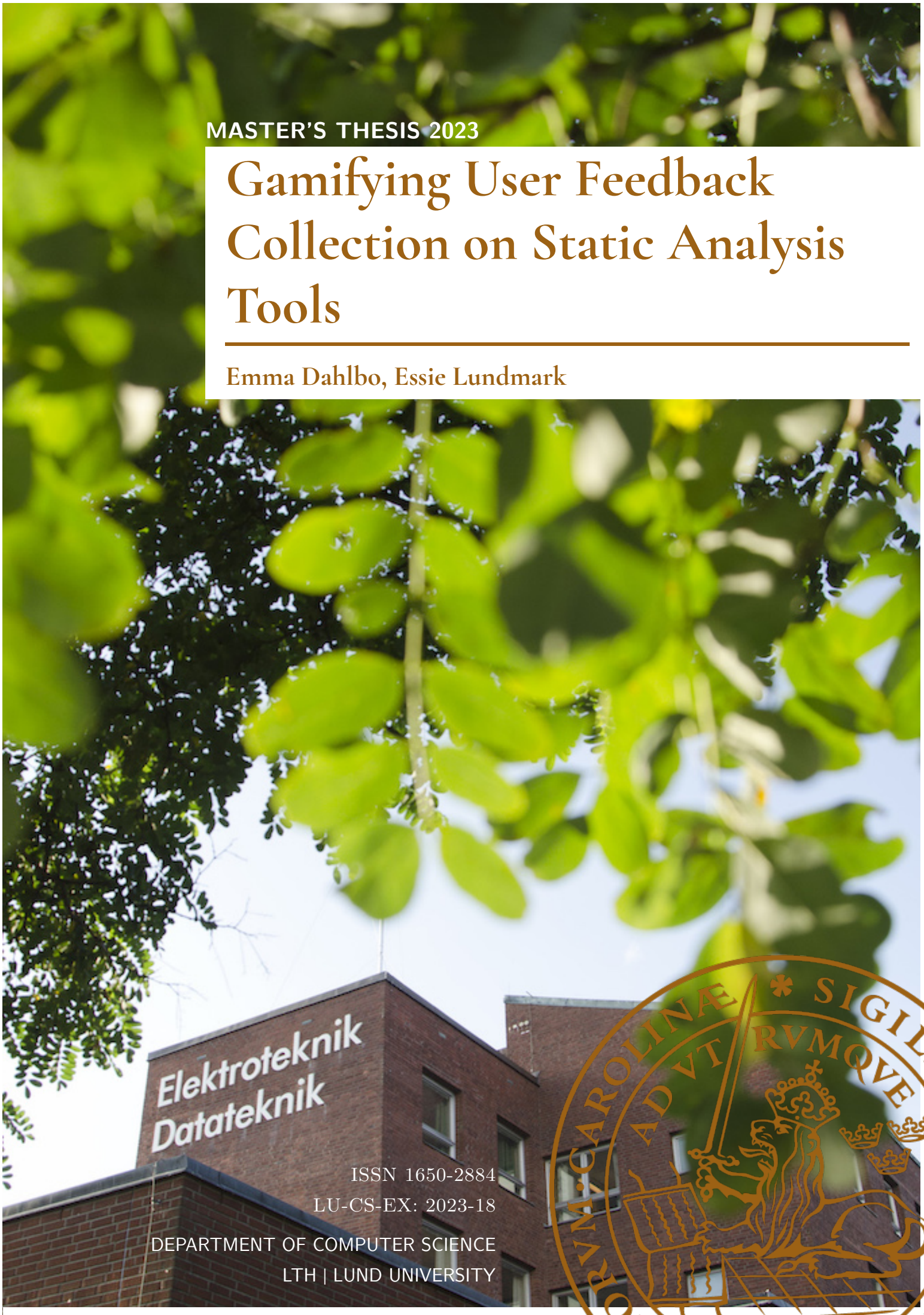


MASTER'S THESIS 2023

Gamifying User Feedback Collection on Static Analysis Tools

Emma Dahlbo, Essie Lundmark

ISSN 1650-2884
LU-CS-EX: 2023-18

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2023-18

Gamifying User Feedback Collection on Static Analysis Tools

Gamifiering av feedbackinsamling om statiska kodanalysverktyg

Emma Dahlbo, Essie Lundmark

Gamifying User Feedback Collection on Static Analysis Tools

Emma Dahlbo
emma.dahlbo.8242@student.lu.se

Essie Lundmark
essie.lundmark.2178@student.lu.se

June 21, 2023

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisors: Emma Söderberg, emma.soderberg@cs.lth.se
Lisa Eneroth, lisa.eneroth@axis.com

Examiner: Martin Höst, martin.host@cs.lth.se

Abstract

Static analysis tools can be highly beneficial in software development, but developers' engagement with these tools is often limited. One method to better understand these tool issues is through user feedback collection. However, this is another activity that often sees low user engagement. A proposed way to increase engagement is by incorporating gamification.

This thesis investigated whether gamification can increase user engagement in feedback collection on analysis tools. First, a literature review and user research at a large tech company examined static analysis usability issues, feedback collection obstructions, and gamification possibilities. Then, a system was developed for giving analysis feedback in Gerrit; it was deployed to around 900 developers over 14 days, with half of the users using a gamification version. Finally, the engagement results for gamification users and the control group were evaluated to determine if gamification is a promising method for increasing static analysis feedback.

Our results show that while gamification did not particularly increase the number of feedback givers, it did increase user engagement among the active users; the gamification group was found to leave 147% more feedback than the control group and have a slightly prolonged system engagement time. These results, however, were not entirely supported by the follow-up interviews, where most participants did not express any particular enthusiasm for the gamification features.

Keywords: feedback collection, gamification, static analysis, user engagement

Acknowledgements

We would wholeheartedly like to thank our supervisor at Axis, Lisa Eneroth, for her continuous support and enthusiasm throughout our entire thesis work and for her impressive ability to find the right people for us to talk to. We would as warmly like to thank our LTH supervisor Emma Söderberg for her invaluable support and input, and our examinator Martin Höst for his guidance.

Furthermore, we want to thank all involved employees at Axis, including everyone that participated in interviews, answered questionnaires, participated in early user tests, and in the final feedback collection and evaluation. We would also like to thank the many employees who helped us with the system design and implementation setup. While they are too many to mention all by name, special thanks go to David Åkerman, Tobias Hagelborn, Evgeni Ivanov, and Rikard Almgren. An honorable mention goes to Stefan KW Andersson for his enthusiasm and valuable input during this project.

Contribution statement

Research and report

Both authors have been equal participants in preparing questions and other material for meetings and interviews. During the interviews, Essie asked the questions while Emma took notes.

Emma was responsible for researching static code analysis and feedback acquisition for our literature review, while Essie was responsible for the gamification research.

Both authors contributed to the initial outline of the thesis, and the initial drafts of chapters and sections were a combination of subsections from both co-authors. Initial drafts where one person had significant responsibility were: 4.1, 4.2, 5.2, 6.2 (Emma); and 3, 4.3, 5.1, 6.1, 6.3 (Essie).

A second draft of the report was produced by Essie, focusing on restructuring to remove irrelevant parts and cohesion: content-wise and language-wise. Again, both authors were involved in proofreading and finalizing the report.

Design and implementation

Both authors have been equal participants in the research phase of the implementation, where different design ideas were researched and discussed; this also applies to the final design of how the different parts should interact. For the technical implementation work, the division was as follows:

- Emma wrote most of the Gerrit plugin components – the frontend UI, the backend REST API and SSH command module, the configuration authentication, separate plugin logging functionality, and documentation – and was responsible for integrating Essie's code into the plugin.
- Essie wrote most of the gamification logic and rules, including the code for gamification logic later incorporated in the Gerrit plugin. She also designed the Confluence space and wrote the game dashboard scripts. She designed and implemented the database and Elastic connections and the Kibana dashboards on the Confluence space. Furthermore, the code in the Jenkins pre-submit job was written by Essie.

Contents

Chapter 1
Introduction

Using static code analysis tools can be highly beneficial in software development, for instance, to verify that the code holds up to security standards and requirements [1]. Meanwhile, previous research shows that developers' engagement with static code analysis often is limited, which can be linked to several usability issues [1], [2]. One method to better understand the issues with static code analysis (SCA) tools and how to handle them is to collect developer tool feedback. However, feedback collection is another task that often sees low user engagement [3], [4], [5], [6]. One possible way to tackle this issue is through *gamification*.

This thesis aims to investigate and evaluate gamification as a way to increase engagement in leaving feedback on static code analysis tools. We conducted the work in the context of SCA tools and processes at Axis Communications, which will be described in Chapter 2.

This introductory chapter begins with the thesis objectives, presenting our research questions and the project's delimitations. Then, we provide an overview of the report and its structure, describe how this thesis contributes to current research, and present a glossary of terms used throughout the report. Last, we review some related work on the subject.

1.1 Objectives

The overall objectives of the thesis were the following: To develop a system for collecting feedback from developers on SCA tools, specifically the vulnerability scan tool Coverity [7]; explore in which ways gamification strategies can be applied to improve the amount of feedback collected; and pinpoint what is hindering engagement with current static security analysis tools at Axis. The last point was achieved through user research and evaluation of the collected feedback. Thus, the thesis has three main themes: usability issues in static code analysis tools, increasing engagement in feedback collection, and evaluating the effects of gamification on feedback collection.

1.1.1 Research questions

Several research questions (RQ1-RQ5) were formulated based on the aforementioned objectives:

- **RQ1:** What does previous research say on the effectiveness of different gamification methods?
- **RQ2:** What does previous research say on usability issues with static analysis tools, and on giving feedback on software like these tools?
- **RQ3:** From a developer's perspective, which aspects hinder interest and engagement with static analysis tools and with giving feedback on these tools?
- **RQ4:** In which ways can gamification be used to increase the engagement with giving feedback on static analysis tools?
- **RQ5:** When evaluated at Axis Communications, does gamification increase interest and engagement for giving feedback on static analysis tools?

1.1.2 Delimitations

The thesis's main focus was to evaluate the effects of gamification on feedback collection. The collected feedback will also provide insight into usability issues of SCA tools and how these can be handled at Axis; this is an objective for the company but will not be the main objective of the thesis per se and will thus not be analyzed in depth.

1.2 Report overview

This report describes the thesis work, from research to system design and final results. After this introduction, an overview is given of the context in which this thesis was conducted, i.e., how Axis works with SCA tools (Chapter 2). Then, our method is described (Chapter 3), followed by a literature review (Chapter (4) of the following relevant topics: SCA tools and practices, feedback acquisition in software, and gamification. This is followed by our user research results (Chapter 5), given by interviews and a questionnaire distributed at Axis. Based on literature and user research results, our feedback system design is presented (Chapter 6), followed by the results of the system evaluation and a small follow-up user research study (Chapter 7). The report is concluded with a discussion around threats to validity (Chapter 8), and a discussion about the results, both user research and system evaluation results (Chapter 9). At the very end of the report, we list the conclusions of this thesis (Chapter 10).

1.3 Contributions

This thesis contributes to current research with the results from a 14 days trial of a gamified feedback acquisition system at a large tech company in Sweden. The gamification design was customized to a work culture where enthusiasm about work quality and efficiency is already highly valued, but previous attempts at feedback acquisition have seen low engagement. Our results contribute to the research on the effectiveness of gamification in the given context. Furthermore, by findings both in our user research and the evaluation of the feedback system, the thesis contributes insights into feedback collection preferences by users in this context.

Additionally, this thesis offers clear and comprehensive instructions on methodology and material for repeating similar research; the method describes all steps from initial research to system design and evaluation. The method is also agnostic to the context in which it is applied.

1.4 Glossary

This section lists a few terms used throughout this report which will be important for the reader's understanding of the background and the system design. This includes short descriptions of some development tools used at Axis.

CI Continuous Integration.

Code review regards manually analyzing a program to ensure that code holds up to set standards.

Confluence [8] is a web-based wiki tool provided by Atlassian, aimed at companies to use for employee collaboration and sharing informational pages.

Coverity [7] is a static code analysis tool focused on security vulnerabilities, supporting 22 different programming languages, e.g., C, C++, and Java.

Coverity issue is a warning by Coverity about some problem in the code.

Elastic Stack [9] is a set of SaaS (*Software as a Service* are cloud-based tools users can access over the internet) tools developed by Elastic NV, consisting of Elasticsearch, Kibana, Beats, and Logstash. This thesis references Elasticsearch, a NoSQL database and search tool, and Kibana, a data visualizer.

Gerrit [10], [11] is a free, web-based code review system where developers can review each other's code before committing to the real repository. Gerrit uses Git for handling version control.

Gerrit change is a commit that has been pushed for review.

Gerrit patchsets are iterations of a commit achieved by amending an existing commit.

Gerrit robot comment is a comment in the code view in Gerrit published by a process rather than a person.

IDE Integrated Development Environment. Examples include Visual Studio Code, Eclipse, and IntelliJ IDEA.

Jenkins [12] is an open-source automation server where projects can be built, tested, and deployed.

SCA Static Code Analysis.

SCA checks are specific options in a SCA tool that look for specific errors in the code.

1.5 Related work

Two previous thesis projects have been made as a collaboration between Lund University and Axis about feedback on static code analysis. During the first thesis [13], the open-source system MEAN (MEta ANalyzer) [14] was developed based on the success factors of Google's code analysis platform Tricorder [15]. The subsequent thesis [16] was conducted to improve the existing MEAN system. Both Tricorder and MEAN partly aim to collect developer feedback on SCA tools. However, they also focus on making immediate changes to the SCA tools based on the provided feedback, which is not considered in this thesis. MEAN was the system that initially inspired and motivated our thesis work.

1.5.1 Tricorder

One of the main inspirations for the development of MEAN was Google's internal static code analysis system Tricorder, as well as five main principles that Sadowski et al. [17] have identified as the key to Tricorder's success at Google:

1. The users should decide what is considered a false positive and have the ability to affect the functionality of the tools.
2. Tricorder heavily encourages user contribution.
3. Usability improvements to the tools are achieved via a data-driven, user-data feedback loop.
4. The tools must be sufficiently integrated into the workflow.
5. Tools should allow customization at a project level, not a user level, which enables teams to work in a more unified way.

Especially point 3, regarding the feedback loop and how this was handled in the MEAN system, is interesting for this thesis.

1.5.2 MEAN

The primary motivation behind MEAN was to see if the Tricorder factors listed above also would work in a different context [14]. MEAN is integrated into the code review tool Gerrit (following the fourth principle) and runs a selection of custom static analysis tools when code is submitted for review – the results of the analyzers are then presented as robot comments in Gerrit. Each robot comment has buttons that allow user feedback to be collected (for example, if a particular comment was *not useful*). This feedback was forwarded to the maintainers of the system, who would tweak which checks were included in the analysis (following the first, second, and third principles) in global and local (repository-specific) configuration files (fifth principle).

The MEAN system was gradually deployed at Axis over 11 weeks and was used by 446 developers across R&D [13].

During the second thesis [16], suggested fixes (generated from the analyzers) for the SCA warnings were added to the robot comments.

User engagement is defined in the MEAN paper [14] as the number of clicks on any robot comment divided by the total number of generated robot comments – this is an important metric for feedback systems like this and Tricorder. With this definition of user engagement, one major success of MEAN is the increased engagement numbers compared to Tricorder – 2.9% (863 clicks for 20,834 generated comments during the deployment time) versus 0.8% (764 clicks for 93,000 generated comments as daily averages).

Chapter 2
Software development at Axis

In this section, we account for some Axis-specific background information that may be useful to the reader as context for this thesis. An overview is given of development, code review, and SCA practices used at Axis, and more specifically, of the Coverity use. An overview is also given of the relevant work culture at Axis.

However, it should be noted that the tool setup at Axis, and the processes around this, vary greatly between different teams and even subteams, sometimes even within teams due to personal preferences. Developers are often unaware of other teams and developers' setups. Thus, a conclusive overview of the development setup cannot be given.

Most information in this chapter has been extracted from the developer interviews (both informal and semi-structured) done during our user research.

2.1 Development processes

Regarding version control and code reviews, most Axis developers use Gerrit; while some use GitHub, this thesis will only focus on Gerrit users. The Gerrit settings differ slightly between teams and repositories, but a typical workflow can be summarized as follows: A commit is created or amended and then pushed for review, creating a new patchset in Gerrit; the patchset will be available in either a new change (with a new change ID) or an existing change (if amended). Several tasks must then be scored and approved to allow code submission. Typically, scores range between -2 (*do not submit*) and +2 (*approved*), and how many votes, and of which kinds, are needed for code approval vary between repositories. Commonly, a +2 vote is required for full approval. Tasks have individual permissions, and users may only have permission to vote -1 or +1 on a -2/+2 task. Figure 2.1 gives an overview of the Axis development workflow.

One of these tasks is called *Code-Review* and is the process of assigned reviewers manually inspecting the code. The needed votes and the number of reviewers differ between different team practices. Other tasks, where automatic processes score the code, can also be configured,

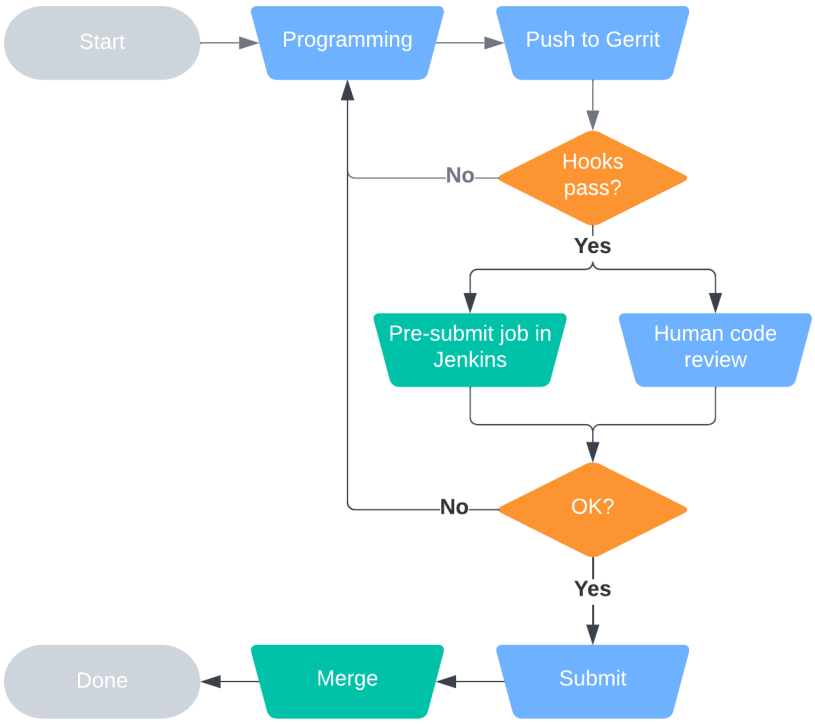


Figure 2.1: Overview of the general development setup at Axis, though not uniform among all teams. Green boxes indicate automatic processes, while blue boxes indicate processes conducted by humans.

often with ratings from -1 to +1. One such automated task is to trigger a Jenkins pre-submit job, which, for example, can build and run tests on the code. The results from the pre-submit job are commonly returned to Gerrit in the form of a comment on the relevant patchset.

Another part of the development environment is git hooks; git hooks automatically trigger a script upon a specific git event. Teams may optionally configure their own git hooks, for example, in order to prevent something from being pushed if the code does not pass some specific check.

2.2 SCA tools practices

This section describes some common SCA tool practices and setup details in Axis; which SCA tools are used by teams and individuals depends on many factors, such as which programming languages the team uses. Much of the Axis code base is written in C and C++, while Python is frequently used for tests. SCA tools are often used for all these languages, for example, type-checking for Python and Coverity for C and C++. Some teams have also recently started to use Rust; in a few teams, this is the main programming language.

2.2.1 Local programming

When it comes to local programming (i.e., the tasks developers do, confined to their local computer [18]), the SCA setup at Axis is very developer-dependent. Many developers mainly use the terminal for development, others use IDEs such as VSCode [19], which may harbor SCA extensions such as Pylint [20] and mypy [21]. In this context, formatting tools, which are a kind of SCA tool, are commonly applied. Some other examples of SCA tools often applied during local programming are clangd [22], PEP 8 [23], clang-tidy [24], ESLint [25], Black [26], and Sparse [27] Also, git hooks are used by some teams for running SCA programs at code pushes; the most common SCA task in this context seems to be code formatting. Notably, at least one team at Axis uses the SCA tool cppcheck [28] as a git hook.

2.2.2 Code review

After local programming, during the code review context, SCA tools are often applied in a different manner. Commonly, one or more SCA jobs or tests are run via automatically triggered Jenkins jobs when a commit is pushed to Gerrit for review. The most notable of these SCA tools is Coverity, which runs on almost all C and C++ code that is pushed to Gerrit.

Coverity is triggered upon patchset-upload and runs from a Jenkins pre-submit job. After analysis completion, the results (passed/failed with issues) are posted in a Gerrit comment at the relevant change. If the analysis failed, the Coverity issues can be viewed by developers in a couple of different ways, via:

- **The Coverity UI**: A web-based tool by Synopsys, where all Coverity issues for whole projects can be viewed.
- **An HTML page**: A web page displaying all current issues in the given repository, in a table format. Each table row has a link to another HTML page, which displays the relevant issue as a stack trace incorporated in the code.
- **A JSON file**: A text file showing more details for the new, local issues, i.e., the ones that were introduced by this patchset.

In addition to posting the analysis results, Coverity also votes on the *Verified* label of the patchset (-1 if failed, +1 if passed). The role of the vote differs depending on repository configurations. In some teams, a -1 vote from Coverity makes code merging impossible before the issues are fixed. Other teams' repositories allow for the developer to ignore a Coverity vote and submit the code anyway, or delete the vote with a written justification. Other teams have fully disabled the Coverity analysis on some repositories.

2.2.3 Frequency of SCA

How often and much the SCA tools are used at Axis varies greatly between teams. Some teams use no other SCA tools than formatting tools and analysis provided by the compiler. Others use a much more extensive SCA-tool setup, and report that SCA is done on all code they write, by continuously running SCA tools in their IDE and triggering SCA on code pushes to Gerrit. Meanwhile, some developers might only run local SCA tools if they think the code might fail the git hooks or Coverity checks.

2.2.4 SCA tool decisions

An important part of the SCA-tool setup is how decisions are made, and by whom, about which tools to use. This also varies between teams. Developers generally have a lot of individual control over their tool use. Developers within teams may have an important influence on if and how SCA tools are introduced into their team setup; this is very much up to the individuals in the team - meaning that teams without individuals with this drive may end up without SCA tools at all.

Meanwhile, git hooks are generally configured on a repository basis, and Jenkins jobs such as Coverity are on a more company-wide level. The use of Coverity is more or less enforced for all code in C or C++, though it has been disabled by certain teams, and the analysis results similarly can be ignored, at least this is possible with some teams' setup. This SCA tool was added externally, outside of the teams' control; some developers do not know who configured Coverity.

2.3 Coverity use at Axis

In this report, the focus will be on the static code analysis tool Coverity [7], a tool distributed by Synopsys, which on Axis is run on almost all C and C++ code.

Due to licensing decisions, Axis currently holds significantly fewer licenses than they have users, and only the license-holding users have access to the Coverity UI. However, all users can view both the HTML file and the JSON file with the Coverity analysis results; although, these files are only available for a limited period of time due to storage management. This workaround with the HTML file was originally created both due to the lack of licenses and to avoid the slow Coverity UI.

2.3.1 History of Coverity at Axis

Coverity was first introduced at Axis in 2018. First, the tool underwent an internal trial and proof-of-concept to check if the tool would be useful, and if it could integrate well enough into the workflow to motivate its use. This was conducted in three phases: trial (week 20 2018), proof of concept (August 2018), and rollout (September 2018 to mid-2019).

In the first part of the trial (week 20 2018), it was found practicable to run Coverity on the entire firmware code base. During this time, the issues found by Coverity in 9 modules were manually triaged by developers, and the result indicated that less than 10% of the reported issues were false positives. The Coverity UI also received positive feedback from developers. The trial conclusion was that Coverity finds real code issues, that the false positive ratio was acceptable, and that the trial should continue with a proof of concept by looking into how well Coverity could integrate into the workflow.

In February 2020, Coverity feedback was collected via interviews with a few developers in different teams. The reported results indicate negative results. The rate of dismissed Coverity issues compared to actual bugs was 68%, which is reported as a false positive ratio (though, there may be threats to the validity of these results, for example, if developers were more inclined to report false positives than to report when warnings were not false positives). The results state that Coverity is not worth the effort for all legacy code, since it takes effort to

go through and triage all the old issues; Coverity is also said to not integrate well with the workflow, even for reporting issues in new code. Instead, it is most useful when run on a select few high-risk components.

In October 2020, another update report was published. At this point, Coverity was utilized by almost all development teams, both when developing new software and for integration and regression testing. It was stated in the report that Coverity finds many issues which would otherwise have been missed and that the tool saves time since developers by themselves do not have to find and verify these issues. Some teams criticized that the UI was slow, but the conclusion of the presentation was that most teams found the tool very valuable.

Since then, the benefit of Coverity has been questioned internally at Axis, and the perception within the Coverity license management team became that the developers generally were opposed to the tool. In November 2022, Axis disabled the use of Coverity in all projects since the Coverity credentials had not been renewed. When this happened, teams objected and claimed that they needed Coverity; eventually, several licenses were put back in place. As pointed out above, there are currently fewer user licenses than users, so not everyone is able to access the UI, though this does not limit how many receive the analysis results.

The team that has been responsible for evaluating Coverity at Axis has voiced concern with the difficulties to understand and optimize the use of this tool, compared to other security practices such as threat modeling and testing. Even though there have been attempts, as described above, the team has experienced a lack of tools for measuring Coverity activities, usage data, and developer opinions. This uncertainty of the tool's worth, and the need for it, is a large motivating factor for this thesis. One aim is to get data points for making better decisions with regard to SCA tools.

2.3.2 Previous Coverity feedback through Gerrit

There was a previous attempt at gathering developer feedback on the Coverity analysis through Gerrit; this feedback mechanism was only available for a limited period of time. However, it was pointed out to us by the implementer that since the feature mimicked the regular Gerrit functionality (with voting -1 and +1 on if the Coverity analysis was helpful or not), the mechanism might have been misunderstood by developers and the validity of the data can therefore be questioned. One theory is that developers interpreted the vote as a vote on whether the Coverity analysis was correctly done or not, which would be similar to the Gerrit-native implementation.

2.3.3 Existing data from Gerrit to Elastic

Axis already collects a large amount of data from Gerrit via an integration between Axis' Elasticsearch instance and Axis' Gerrit instance. Via this integration, Elasticsearch collects data about all Gerrit events, such as commits and comments added. Since each Coverity result in Gerrit is presented in a Gerrit comment and a *Verified* vote on the patchset, the results history (pass/fail of Coverity) can be extracted from Elasticsearch. This has previously been used for presenting data on Coverity usage and results on a company level. Figure 2.2 shows an example of a graph that can be extracted from the Coverity data already collected in Axis' Elasticsearch.

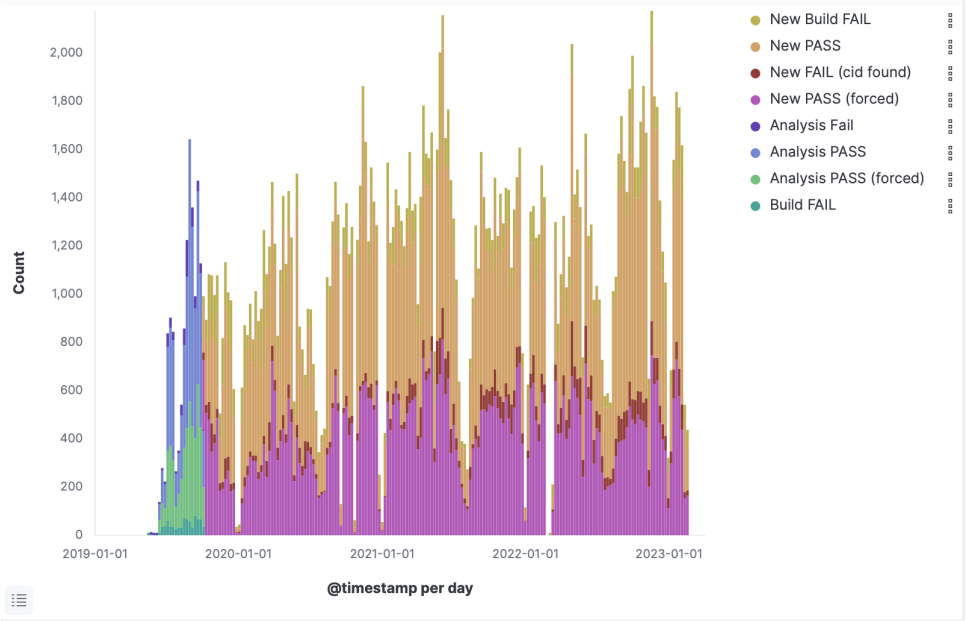


Figure 2.2: Graph on Coverity results data collected in Axis' Elasticsearch instance.

2.4 Work culture at Axis

This section gives a short overview of the work culture at Axis since this will be important for designing the user research method and for interpreting the results. The information here is based on public company information from 2023, as well as input from our supervisor and her colleagues at Axis.

Axis Communications was founded in 1984 and is mainly known for developing network camera solutions, though they also have other network technology such as door stations and speaker solutions [29]. The company is mainly concentrated in one large head office in Lund, Sweden, where there are around 3,000 employees (including consultants, which is only a small minority) as of May 2023; the whole company includes around 4,000 employees in total. While there are many other Axis offices, in around 50 different countries, especially developer employees mainly reside in the Lund office. These developers are parted into different departments and teams, where each team and even subteam is to a large degree self-managed with a lot of autonomy.

At Axis, activities "for fun" during working hours are typically very encouraged, for example organizing "Lucia-tåg" and Christmas performances. Another indication of this rather relaxed work culture is the tendency for teams to give themselves playful team names, inspired by comic books or similar references.

Employees log their own hours according to a flextime principle, and typically no time stamping is used. At least in the R&D department, where the majority of our test group participants are based, the salary is performance-based. However, also contributing to fun activities as those mentioned above can typically be brought up during salary discussions as an indication of engagement in the workplace.

2.4.1 Advent of code

One recurring event at Axis is the company-wide participation in Advent of Code, a coding problem-solving competition by Wastl, held as an advent calendar each year. It is voluntary for Axis employees to participate, and is not something done during working hours. Still, each year there is an Axis leaderboard for competitors, and rewards such as T-shirts are handed out after the competition. This has been held for the last two years, with a large increase in interest the last time (December 2022), and it is expected to increase also this following December. The first year, there were two separate Axis leaderboards, one internal (only for Axis users) and one external; the last year, there was one single leaderboard for all users, where 233 employees and 177 non-employees participated. Of these, 371 got at least one point, i.e., actually submitted some successful solution in an advent of code challenge.

Chapter 3
Method

In this chapter, we describe the method used in this thesis project, which consisted of the following subsequent steps: a literature review to gather sufficient foundational knowledge on the subject; user research; system design; and evaluation. A basic overview of the different parts of the method is visualized in Figure 3.1.

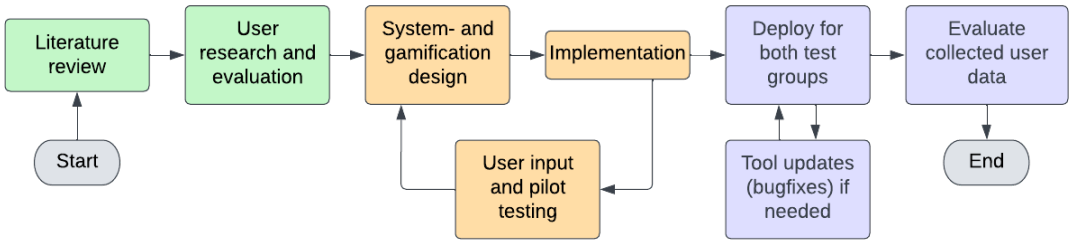


Figure 3.1: An overview of the different project steps and in which order they were conducted.

There exist many different process frameworks and recommendations for designing and evaluating gamification systems. In this thesis, we used a process inspired by some of the generic frameworks found in previous works, especially the ones presented in the framework literature review by Mora et al. [30]. Below, the general process is described; some of these steps are later expanded on in sections of this chapter.

1. **Literature study**: This was conducted to gather sufficient foundational knowledge for the following steps and to answer RQ1 (*What does previous research say on the effectiveness of different gamification methods?*) and RQ2 (*What does previous research say on usability issues with static analysis tools, and on giving feedback on software like these tools?*).

2. **User research**: This aimed to answer RQ3 (*From a developer's perspective, which aspects hinder interest and engagement with static analysis tools and with giving feedback on these*

tools?), and, together with the literature study, provided a basis for the design of the feedback system. In short, the main goal was to examine developers' motivational drives for using and giving feedback on SCA tools, as well as developers' opinions on gamification. We also studied the parts of the company culture that may be relevant to how gamification can be applied, as encouraged by Werbach and Hunter, cited in Dal Sasso et al. [31].

3. **Analyze the data and plan**: We concertized the current SCA tool usability issues and issues with collecting feedback and specified reasons and desired outcomes for the gamification of a feedback system. We also defined moral principles and ethical values that were important for the given gamification context, as recommended by Versteeg according to Mora et al. [30].

4. **Design the overall system and the gamification features**: We decided which game principles and elements would be used, based on the user research results, including the distribution of player types. The system design was also based on the previous literature study and user research. This examined different possibilities regarding RQ4 (*In which ways can gamification be used to increase the engagement with giving feedback on static analysis tools?*).

5. **Plan the evaluation**: At this point, we defined fixed quality parameters to measure the desired outcomes and possible other outcomes.

6. **Implement the system**: Two versions of the system were developed: One with gamification and one without.

7. **Pilot-test the design**: Early user testing is recommended, for example, by Morschheuser et al. [32]; since time did not permit us to develop Lo-Fi and Hi-Fi prototypes, we settled on some limited pilot testing before deployment. At the same time, we also tested the system against cheating and rule-bending, as also recommended by the same paper.

8. **Deploy and test**: We tested the system in two different, parallel versions as recommended by Hamari et al. [33]. The system was tested for two weeks for all Axis Gerrit users by dividing them into one gamification group and one control group, each testing a different version.

9. **Evaluate the system**, with the primary goal to provide an answer for RQ5 (*When evaluated at Axis Communications, does gamification increase interest and engagement for giving feedback on static analysis tools?*). According to Monteiro et al. [34], gamification evaluation should use a mix of qualitative and quantitative data collection and analysis, with both subjective and objective inputs. In accordance with this, we analyzed data collected automatically by the tool and conducted a few chat interviews to nuance our findings.

As recommended in Morschheuser et al. [32], gamification processes should preferably be iterative with many progressive improvements and user evaluations. However, because of the thesis time restrictions, only one major design process iteration could be achieved.

3.1 Literature review

The initial literature review was conducted as three separate endeavors: SCA tool usability issues, feedback collection methods, gamification solutions, and previous gamification results. Though similar methods were used for all three studies, they are described separately for clarity.

3.1.1 Static code analysis theory

Regarding previous research on SCA tool usability issues, an initial set of resources ([1], [2], [14], [15]) was provided by the supervisors of this thesis project. From there, we used forward and backward snowballing to find other relevant sources. We also used keyword-based searches on Scopus, using mainly variations of *static code analysis*/*code analysis*/*static analysis*/*SCA*, and *+usability*. In some searches, *bugs* was added to filter out outside sources that did not regard software analysis.

The final sources included in the literature review were chosen either based on being well-cited or recently published, with relevant topics.

3.1.2 Feedback theory

For this part of the literature review, we used a combination of keyword-based searches on Google Scholar and forward and backward snowballing wherever relevant. In this way, we found well-established and referenced studies, as well as more specific studies for applying to our research. All keyword-based searches started with the phrase *feedback acquisition*.

3.1.3 Gamification theory

To gather information about gamification and the current state of gamification research, an initial broad search was done, and the results were then narrowed down step by step to get more specific references about software engineering and feedback collection gamification. Overall, we used multiple different search phrases and keyword combinations to avoid relevant studies being omitted. Searches were done on Google Scholar, Scopus, and LUBSearch with similar search terms, of which a few are accounted for below. On occasion, we additionally used backward snowballing, for example, in Hamari's literature review paper [33], where we picked out the most relevant studies for consideration as references.

The initial broad search term on Google Scholar, which resulted in 18500 hits, was:

```
(gamif* OR gameful) AND ("software engineering" OR ~development
OR "static code analysis" OR "software test*")
```

The much more specific search term on Google Scholar, which gave only 35 results, was:

```
allintitle: (gamification OR gamifying OR gameful)
AND ("software engineering" OR "software development"
OR "static code analysis" OR "software test*")
-education -learning -training -course
```

Variants of this term were also tried, for example, to find information about gamification in SCA tools, which was not a very common research topic.

Throughout the search, the results were narrowed down by disregarded papers with one or multiple of the following traits:

- Focused on specific gamification contexts irrelevant to us (health, education, and similar).
- More focused on serious games or real full-fledged games than gamification.
- Could not be accessed by us without buying additional database rights.

We also aimed to mainly use peer-reviewed papers (journals or conference articles, for example, from IEEE, that guarantee peer-review). However, this proved difficult because of the relatively small amount of relevant research.

3.2 User research

This section describes how the user research was conducted and how the results were evaluated. A mix of interviews and questionnaires was conducted during February and March 2023. First, four initial informal interviews were held, and the main data gathering consisted of eight more thorough interviews and a questionnaire. We used both interviews and questionnaires to triangulate our data; however, because of time limitations, we used both methods in parallel instead of using the results from one study for designing the next.

Note that all questionnaires were in English, but all interviews were held in Swedish since all interview participants happened to speak Swedish. This means that all interview quotes are translations.

3.2.1 Goals of user research

The general goal of the user research was to address RQ3 (*From a developer's perspective, which aspects hinder interest and engagement with static analysis tools and with giving feedback on these tools?*) and RQ4 (*In which ways can gamification be used to increase the engagement with giving feedback on static analysis tools?*). After having conducted the literature review and the initial informal interviews, we formulated the following more fine-grained goals for our main user research:

- Get insight into the current SCA tool usage at Axis. Which experience with SCA tools do developers have, how do they use them, and how are they integrated into the workflow? This was important to know as a basis for our system design.
- Get insight into developers' general experience with SCA tools and SCA tool warnings, developers' motivation to use the tool and fix warnings, likes/dislikes, and contributing factors to this.
- Get insight into developers' motivation for leaving feedback on SCA tools. How motivated do they think they would be to leave feedback, have there been previous possibilities to leave feedback, and what would motivate them to do it?

- Get insight into demographic data, such as age distribution, sex, and tenure in the intended users. This was a goal to understand the company culture better.
- Examine developers' preferred gamification elements and the differences in preferences across the company. This was done by examining developers' Hexad player types [35] and asking for opinions on some specific gamification elements, which were deemed likely to provoke strong opinions, during the interviews.
- Get insight into developers' thoughts on sharing data, especially gamification data. This was to better design the gamification system in a moral and considerate way that could appeal to as many users as possible.

3.2.2 Initial informal interviews

As a first step in the user research, we held four informal discussions with different developers and team representatives at Axis (five participants in total since one interview was with two persons simultaneously). This was both to get a better overview of the development setup and to plan in more detail what to ask in the following more thorough interviews. These interviews were conducted as 30-minute informal discussions with open questions. The interview subjects received a short description of our thesis and questions before the meeting. Things discussed included the team structure and development setup, the team's general attitude toward SCA tools and their use of these tools, and their general attitude and previous experiences with gamification at work.

3.2.3 Semi-structured interviews

Eight more thorough interviews were conducted, with a semi-structured interview approach to get the benefits from both the structured nature, while also being able to ask further questions to get more qualitative data.

User sampling

In order to find suitable interview subjects, both for the initial interviews and the semi-structured interviews, we reached out to developers at Axis, whom our supervisor and her contacts guessed would have opinions on the subject. Additional subjects were then found via snowball sampling by encouraging the interview subjects and others who declined to ask colleagues if they wanted to partake; notably, one interviewee collected a list of team members who would be willing to participate, of whom we interviewed two. For the sampling, the only restriction was that the subjects should work with or recently had worked with development at Axis and use Gerrit as their code review tool. We also tried to include different types of teams and sub-teams from different company departments. This, and the small number of restrictions, was done to get a broad picture of the company culture.

We also sent out a form with the questionnaire as a name gathering for people interested in helping out. Of the seven people who sent their names as a notice of interest, two were asked for interviews (since they seemed to complement the previous interviewees), and one responded and participated.

Table 3.1: A summary of the tenure and roles of the interview subjects. The given values are estimations made by the interviewees, rounded to whole years.

ID	Years at Axis	Years as developer
I1	3	3
I2	10	10
I3	2	5
I4	1	1
I5	3	15
I6	25	25
I7	4	4
I8	6	6

Interview participants

The eight chosen interviewees were from two different departments in Axis, the R&D department and the New Business department, and within these departments, the interviewees were scattered between different teams. All interviewees currently had some part of development tasks, but multiple also had additional roles such as scrum masters, architects, and security leads, though they previously worked with more development-focused tasks at Axis.

A summary of the interviewees' roles and tenure is presented in Table 3.1. All interviewees were men. The listed interviewee IDs will be used for references in the user result section, Section 5.1.

There was an overlap between people that participated in the initial informal interviews and those that participated in the semi-structured interviews since all the initial five interviewees were asked to participate again. There is also an overlap of people participating in the final pilot testing, described in Section 3.3.2, and the chat interviews described in Section 3.4.4. This overlap is visualized in Figure 3.2.

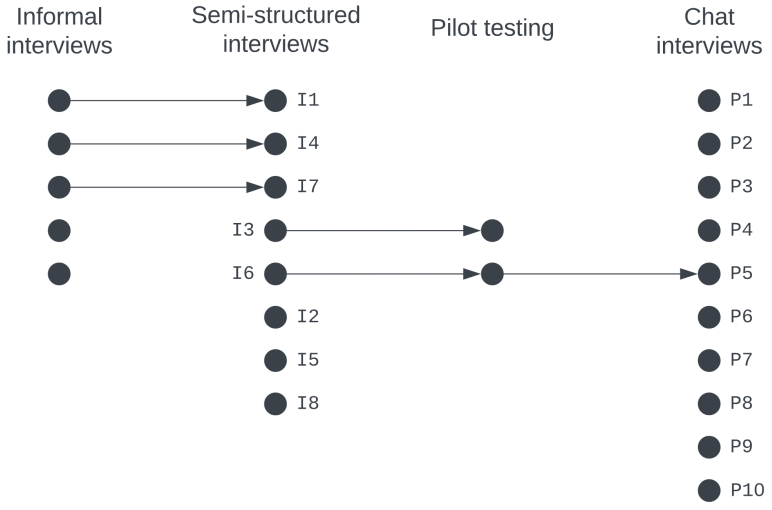


Figure 3.2: The overlap of participants in different steps of this project's user data gathering.

Interview structure

The interviews were estimated to take 30-45 minutes and were scheduled as 45-minute meetings. The mean interview time was 32 minutes. During this time, one of the authors asked questions while the other took notes and added clarifying questions if needed. The interviews were also recorded after informed consent. Afterward, the recordings were listened through, and the notes were supplemented into transcribed data, though not in extreme detail, for example, regarding grammar, repetitions, voice changes, and body language. The recordings were occasionally revisited later for some clarifications and extracting of quotes.

The questions asked were semi-structured and open-ended, encouraging reflection by the interviewees. The discussion revolved around a few main areas: background information (e.g., team and development setup), SCA tool experience, attitudes toward gamification, and thoughts about sharing gamification data. The interviewer followed an interview protocol found in Appendix B, though follow-up questions were asked, and the question order was changed when found appropriate based on the interviewee's previous answers.

Before the interview, the interviewee was informed about their rights and data handling, and all present signed an interview consent form. The form can be found in Appendix A.

Thematic data analysis

The interview transcriptions and recordings were analyzed with a thematic analysis in six phases, as described by Braun and Clarke [36], though in a slightly lighter variant, to find themes and subthemes in the collected data and to extract the information relevant to our research questions (mainly RQ3 (*From a developer's perspective, which aspects hinder interest and engagement with static analysis tools and with giving feedback on these tools?*). This method was chosen because of its flexibility and because it is a well-known and often-used approach for interview data.

More specifically, we used an inductive bottom-up approach by grouping codes and themes by what we found in the data rather than previously defining groups and themes. This was done to lessen bias and not to miss anything brought up in the interviews. (However, since the questions, in a way, were deductive, to debunk or support certain theories on, for example, SCA tools usage, parts of the analysis can be viewed as deductive as well.)

Below is a list of the six phases of thematic analysis according to Braun and Clarke [36], as we adapted them for our interview analysis:

1. We **familiarized ourselves with the data** by reading through it multiple times. The note-taking author did this by listening through the recordings afterward and adding possibly missed parts in the transcription notes. During this phase, we also reflected on possible assumptions made by the interviewees and noted these informally in the interview notes. The recordings were checked where clarifications were needed.

   Then, both authors read through the notes again, marked the interesting parts, and annotated, for our convenience, how data could be extracted from these parts.

2. We **created the initial *codes***, i.e., the building blocks of subthemes and themes, which groups data or interpretations of data. An example code could be "irritation over false positives". Each transcript chunk may have multiple or no codes. The codes were created by once again reading through the transcripts and the previous annotations and marking chunks with suitable low-level codes.

3. We started **looking for themes** by grouping codes first into a few subthemes and then grouping subthemes into themes. We included a misc theme for codes and subthemes that were difficult to group.

4. We **reviewed potential themes** and revised them multiple times, with both authors considering and discussing the transcriptions as well as the theme set.

5. We did the final **definition and naming of themes** and created a thematic map over the themes to provide a clearer overview.

6. We **produced the analysis report** in the result Section 5.1.

3.2.4 Questionnaire

In parallel with the interviews, a questionnaire (Google Forms) was sent out internally in Axis; it was open for 15 days (nine business days) and received 63 answers. The main goals of the questionnaire were to examine developers' feedback-leaving preferences (what motivates developers to leave feedback) and to determine the composition of gamification player types among Axis developers. This was done both to see the variety in individual preferences and to map the overall company culture at Axis. The player types were examined by using the Hexad framework, which is described in Section 4.3.

User sampling

The questionnaire was distributed in Axis by multiple channels to reach as many employees as possible: The Coverity Teams channel, sent to almost all previous interview subjects and encouraged them to forward to their teams (i.e., a snowballing method) and emailed by our supervisor to people that could spread the survey further. For example, one of our supervisor's contacts sent the questionnaire to the entire Tech department email list. Because of this method, the questionnaire likely reached a lot of employees; it is impossible to tell how many, and thus difficult to estimate the response rate.

The questionnaire was aimed only at Axis developers, which was clarified both in the info text and in the first questionnaire question. Still, there is no way to control that only developers answered. No other user sampling restrictions were used.

Questionnaire format

The questionnaire was parted into three sections: One for the control variables age, sex, and tenure; one for determining the user's Hexad player types; and one for examining the user's feedback-leaving preferences, i.e., their motivation to leave feedback in different circumstances. The questions are listed in Appendix C. All the questions in this survey are answered on a 7-point Likert scale, in random order, with four questions for each player type. The user's score for each player type is then calculated by adding the scores for the four relevant questions.

The method of Hexad player types was chosen since it has been evaluated as effective by Tondello et al. [37], and because most other player types questionnaires, for example, the

classic Bartle [38] player types and the player types by Yee et al. [39], are based on real, complete games and not gamification. We determined that it was preferable to use a player-type framework over only directly asking users which gamification elements they would prefer since users might not know this terminology, have different associations with the terms, or not know their preferences before trying; this is in line with the arguments given for player types by Tondello et al. [37].

The users were offered the choice of entering their email in the form and thus getting an email with their player-type results. This was used as a technique to increase the response rate since it might have been an incentive for people to respond to the questionnaire.

Note that the part of the questionnaire regarding feedback-leaving preferences was not mandatory for respondents to answer and thus might have received fewer responses. This questionnaire design was chosen since not every developer might have previous experiences with SCA tools, and the questions directly regarded leaving feedback on SCA tools.

Data analysis

The questionnaire was analyzed by studying the collected data in Google Forms and creating visualizations via that tool. The player types and player type ratios were calculated according to the Hexad framework [35], [37].

3.3 Design and implementation

During the design and implementation phase, we designed the system and the gamification elements while implementing it. We got continuous feedback from our supervisors and a few other Axis employees, mainly regarding the technical system design and the possibilities for this.

3.3.1 Design steps

During the initial design of the system, a few steps were taken based on the user research data and the insights given from the literature review: We specified the reasons for gamification in our context and the desired outcomes of gamification; we defined moral principles and ethical values important for the given gamification context; we decided on which game principles and game elements should be used in the gamification design; and we decided on fixed quality parameters to use as metrics during the later system evaluation.

In previous literature, a few different gamification design frameworks and recommendations exist, some general and some quite specific. We considered adapting to a particular framework but chose not to since most frameworks (examined in the literature review in Section 4.3.8) were found either too general (and thus not very useful) or too specific and therefore not applicable to our context.

3.3.2 Testing the implementation

Before deployment, the implementation was tested during many iterations, first locally and then on the Axis staging environment. The testing involved the authors doing manual system

tests, including as many use cases as possible. Each time errors or issues were found during this testing, the system was revised, and the implementation was updated.

Pilot tests

Just before deployment, we also conducted a small pilot test with two users, where each user tested the system for around 45 minutes while thinking aloud and showing their screen to the authors. The tests consisted of several use cases given to test subjects, who then tried to follow the use cases in the staging area. We also asked the participants to "try to cheat the system" so that the system was tested against cheating and rule-bending (or at least so that insight was given into the possible ways to cheat), as recommended by Morschheuser et al. [32].

The two developers that partook in the pilot testing were found from previous contacts, and the overlap to previous user research subjects can be seen in Figure 3.2.

After the pilot test, minor updates were made to the design after the insights from this test. Notably, the final game constants (how easy it is to get rewards and similar) were based on insights from the pilot testers.

3.4 System deployment and evaluation

The feedback system was deployed for two weeks, during May and June 2023, for all Gerrit users at Axis. It was then evaluated by analyzing the collected feedback and usage statistics and conducting small chat interviews over Teams with a few selected participants.

3.4.1 Deployment setup

The testing phase, during which the feedback system was deployed, was conducted over 14 days (nine business days). The feedback system consisted of two visible parts to users (see 6.1 for more details): the possibility to leave feedback via Gerrit and view feedback and game data in Confluence. The feedback feature was deployed to Axis Gerrit and thus could be used by any Gerrit user but was only visible on changes where Coverity analysis had been run.

The testing phase lasted relatively short because of time constraints and because the system was never intended to be run for a long time but rather as a short-term feedback-gathering system for getting data for later evaluation.

3.4.2 Test groups

Before deployment, the relevant users for our test groups were extracted by finding all users that, during the previous three months, had triggered or taken part in any Gerrit event at Axis. This group consisted of 929 users, which we randomly divided into two groups: gamification users (464 users) and control group users (465 users). Members of both groups used the same feedback system, but the game features were only used by and visible to users in the gamification group.

There was no guarantee that a user in our test groups was still an active Gerrit user, or that the user regularly interacted with Coverity analyses, as only a subset of Axis Gerrit users use

Coverity in their repositories. However, we chose to make the feedback system visible for all Gerrit users, in order to collect as much feedback as possible, and because there was no prior foolproof way of knowing which Gerrit users interacted with Coverity and not. Since the division between the gamification group and control group was made randomly, we hoped there would be a somewhat equal number of relevant users in each group.

After the experiment, we followed up on the test group distributions. This was examined in an approximate manner, by checking which users, during the system deployment time, had been active (by triggering or partaking in a Gerrit event) in any repository where our feedback system was interactable for at least one Gerrit change (i.e. where it was possible to leave feedback). These users were more likely (though they may not be exclusive) to have visited Gerrit changes where the feedback form was present. Note that they must not necessarily have visited such changes, even though they were active in repositories where such changes existed. Note also that also other users are likely to have interacted with other parts of the feedback system (our Confluence space) since most of our originally intended user base received information about the system.

The results of the examination show that there were 161 repositories in which our system was interactable for at least one Gerrit change. In these repositories, 302 Gerrit users were active during system deployment time. Of the users to engage with our feedback system, all were part of this group of 302 users; thus, this could be said to have been our effective test group. Of the 302 users, 139 could be found in our gamification group, and 159 in our control group - 4 users were in none of these groups, and would automatically have been handled as control group users by the system. This means that the effective test groups consisted of 139 gamification users and 163 control group users. (Note that all users that actually engaged with our feedback system were present either in our original gamification group or original control group.)

3.4.3 Announcements

When the system was initially deployed, two methods were used to make users notice the system and to explain its use: an email was sent to the whole user group (though around 70 of the addresses could not receive emails), and there were announcements made in two different channels in Microsoft Teams that were theorized to reach a large, general fraction of Axis developers.

After the system had been deployed around nine days (six working days), a reminder email was sent to the same email list. This was done since the gathered feedback amount was very low, and it was theorized that many users had not noticed the system's existence.

3.4.4 Evaluation of the system approach

After the testing phase weeks, we evaluated the feedback system and the gamification approach by analyzing the collected data and usage statistics and by examining users' viewpoints with the help of a few semi-structured chat interviews. This section first goes through the metrics the analysis aimed to measure and then describes how the chat interviews were conducted.

Metrics

Previous gamification literature offers some different suggested evaluation methods, though their effectiveness will depend on what is measured, and at present, there does not seem to be a reliable, frequently used gamification evaluation framework [34]. However, the most common evaluation criteria are engagement, performance, satisfaction, and motivation [34] – we will focus on most of these.

Based on our research questions, we decided on a few fixed-quality parameters to measure our tool, which is given in the list below. Each of these parameters was evaluated for both the gamification and control groups.

User engagement : How much users engaged with the feedback system features, for example, by providing feedback or visiting the Confluence space. This is measured by metrics such as the amount of collected feedback, the number of active users, and the form completeness.

User satisfaction : How satisfied the users were with the feedback system.

User motivation : How motivated developers were to use the feedback system.

Please note that this definition of user engagement includes additional metrics than in the MEAN paper [14]. From this point onward, we will refer to user engagement as defined above unless explicitly stated otherwise.

While user engagement will mainly be evaluated with quantitative user data from the system, satisfaction and motivation will primarily be assessed by the chat interviews since these can provide more thorough insights into how developers reasoned during their system use.

It is important to note the difficulty of clearly defining engagement, satisfaction, and motivation, mainly if they are meant to map to similar terms as previous research to allow comparisons. Previous research uses different ways of measuring these aspects. Notably, critique has been voiced that user engagement has been used to mean user participation, though they are different things; a user might show up to work or participate in a gamification platform without being engaged in it [40]. However, our measure of engagement will show users actively taking part in the feedback system since leaving feedback or visiting the Confluence page is not something developers need to do in their usual workflow.

Chat interviews

Ten chat interviews were conducted to evaluate users' satisfaction with and motivation to use the system. We contacted 16 users over Microsoft Teams, with a few written questions, as presented in Appendix D. In a semi-structured manner, follow-up questions were then asked in the same chat depending on the given answers to provide more deep insights and to get clarifications.

We aimed to contact the same amount of gamification users as control group users. We created two groups of the same size: people who had already participated in our earlier interviews or shown interest in helping with the project during the questionnaire distribution (first group); and people who had shown engagement with the project by leaving feedback (second group).

Table 3.2: Chat interviewees by ID and test group, and whether they had expressed a prior interest in our project through participation or the contact form.

ID	Group	Prior interest
P1	Gamification	No
P2	Gamification	No
P3	Gamification	Yes
P4	Control	No
P5	Control	Yes
P6	Gamification	No
P7	Control	No
P8	Gamification	No
P9	Control	No
P10	Control	No

In the first group, we prioritized people we had already been in contact with through interviews or testing, and supplemented with others (randomized) from the contact form to get evenly-sized groups. In the end, we had two people we had been in contact with from the gamification group, four from the control group, and two people from the contact form gamification group.

In the second group, we randomized some but additionally chose people who would complement each other; some people were selected because they had submitted many forms, some because they had gotten many points, and others because they had provided very little feedback (yet still feedback). Due to the nature of the split, the first group contained primarily people who had left little to no feedback. In contrast, the other group was explicitly chosen to include people who had been at least semi-actively participating – with this distribution; we hoped to get a good representation from most of the user base.

Of the 16 contacted users, twelve answered and ten participated in the interviews; more details about the participants can be found in Table 3.2. The discrepancy between answers and participants can be explained by two people not noticing any changes in Gerrit, and thus we assume that none of their projects were included in the testing.

The response rates between the two selection groups varied (see “Prior interest” in Figure 3.2). The first group (“Prior interest = Yes”) had a response rate of only 25%, compared to the second group (“Prior interest = No”) which had a response rate of 100%.

Chapter 4
Theory

This chapter examines previous research on three topics: static code analysis tools and common usability issues with these, feedback acquisition on software systems, and gamification.

4.1 Static code analysis

Program analysis is the process of analyzing code to understand it better. The scope of program analysis is broad, including finding problems such as general bugs, security vulnerabilities, memory handling issues, and design issues, in order to help developers write quality code. There are two main types of program analysis: static analysis, where code can be analyzed without being executed, and dynamic analysis, which requires execution. Dynamic code analysis often occurs during testing, while static code analysis is helpful in supporting developers early in the development process [1], [41].

Static code analysis (SCA) can be conducted by humans via code review and can be automated with SCA tools, which often is a both faster and cheaper option [1]. SCA tools can also help detect issues early in the development process [2]. Despite this, research shows that engagement with SCA tools is low [1], and researchers often struggle with achieving long-lasting use among many users [15].

This section expands on what SCA tools are and how they benefit developers to provide context for this thesis and to aid the feedback system design. We outline previous research on factors that hinder SCA tool adoption and developers' wishes about SCA tool functionality.

4.1.1 Overview

There is a great variety of static code analysis tools and their use cases - from linters (e.g., ESLint) to bug finders (e.g., FindBugs) to tools that run more complex security scans (e.g., Coverity). This section outlines a short, general overview of SCA tools and their usage for readers unfamiliar with the topic.

Warnings

Commonly, SCA tools identify and present code problems of different types to the user. Such presented flaws will, in this thesis, be referred to as *warnings*; other literature may refer to them as alerts. A warning which is falsely reported, i.e., does not represent an actual error in the code, is called a *false positive*.

Usage

A SCA tool may integrate into any part of the development process, and examples of integration points include the IDE, code review, and continuous integration pipelines.

The participants in the 20 interviews conducted by Johnson et al. [1] said that they find value in using SCA tools both during the early stages of development to find newly created bugs and also later – however, they do not mention how many participants held this viewpoint. Vassallo et al. [18] found that while developers found SCA tools generally useful during the entirety of the development process, they tended to focus on different flaws depending on the context they were in at the moment.

This may also depend on the runtime of the tools since some SCA tools (also depending on code size) may execute in a moment while others may even have to run overnight.

Vassallo et al. [18] also presented three so-called development contexts in which SCA tools are used differently. The contexts are local programming, where developers write the code, for example, in an IDE or text editor, and where SCA tools are often integrated as plugins; code review, where SCA is done manually, often with the help of checklists, coding standards, and SCA tool warning reports; and continuous integration, where code committed by developers is automatically built and tested, and where SCA tools can be run automatically as a part of the pipeline, to ensure certain code quality and standard.

Motivations for use

During this literature study, we have found that motivations for using SCA tools generally fall into one of two categories: technical motivations, i.e., arguments about why SCA improves quality in some way, and subjective motivations, i.e., reasons that developers themselves give for using SCA tools.

From a technical standpoint, code analysis can help detect various flaws in the code that, if not fixed, could lead to vulnerabilities. In this way, SCA tools contribute to system security. Many flaws checked by SCA may seem unrelated to security at first glance but often relate to security in the long run. An example of this is bad coding style; it may lead to low readability and understanding, which can become a security concern if code maintainers unintentionally introduce vulnerabilities due to misinterpretations. Research also shows that the same kinds of flaws are introduced into code repeatedly, despite being generally understood and having existing solutions [41]. Tahaei et al. [41] suggest that this is due to a lack of awareness and/or support for developers, something that SCA tools ideally solve.

From the developers' point of view, SCA tools are useful for helping them code better; Do et al. [42] found that 78.3% of the participants in their survey (87 developers at Software AG) motivated their use of SCA tools this way. More specifically, reasons that have been mentioned as developer motivation for SCA tool usage are: to detect security-related code issues [43], [41], [42], assist with using best practices [43], [41], and ensuring correct coding

style [41]. Studies found that developers also care about discovering concurrency-related issues, code performance, and memory consumption [43], [42]. In the same study by Do et al. [42], 21.7% said that they also use these tools because it makes the coding process quicker. On a smaller scale, company policy was also found to be a motivating factor for SCA tool use. In Do et al. [42], 30% of participants said that their use was in part motivated by company policies – the company where the survey was conducted heavily encourages the use of SCA tools, which most likely contributed to this number.

4.1.2 Tool-developer interaction

This section gives an overview of previous research on the interaction between developers and SCA tools, focusing on how developers understand, prioritize and fix SCA warnings.

Understanding warnings

In order to properly prioritize and fix warnings, they must first be understood. Do et al. [42] found that only 36.1% of investigated warnings were sufficiently understood on average, though the results varied between 0% and 80% between developers. The three main reasons for this were noted as the following: the developer being unfamiliar with the reported issue and unsure how to fix it without breaking other areas of the code; the warning explanation from the SCA tool being perceived as unclear; and that the warning covered too much code so that fixing it would need changes in many different places in the code base.

When developers do not understand a warning, Do et al. [42] found that the most frequent behaviors are to leave the warning for later or to ask for help. Sometimes, a warning is also ignored. Things such as researching and fixing the bug or suppressing the warning were only done by less than 10% of developers in the Do et al. study [42].

Prioritizing warnings

Since the number of warnings developers are tasked with is generally relatively high, they often have to prioritize which warnings to fix. Do et al. [42] found in observations that how many warnings are looked into varied greatly between developers, from 20% to 100% of the warnings (in general 65.1%).

Regarding how warnings are prioritized, both Vassallo et al. [18] and Do et al. [42] found that the severity of the warning was one of the most important determining factors. Do et al. [42] also found that developers, to roughly the same amount, prioritize warnings by whether they affect their own code (as opposed to warnings introduced in others' code).

Vassallo et al. [18] also argue that which warnings developers pay attention to is affected by development context, i.e., if SCA is applied during local programming, code review, or continuous integration, for example, during the local programming context, developers may focus more on warnings relating to code structure, style convention, and redundancies, compared to an additional focus during code review on naming conventions and simplifications. Still, the results from Vassallo et al. [18] suggest that developers themselves are not fully aware that they are doing this context-based prioritization.

4.1.3 Usability

In addition to finding bugs, SCA tools are meant to provide good usability and help developers solve the found issues by adequately explaining and presenting them [44]. However, SCA tools have often generally been deemed to have various usability issues, which have made the tools less popular among developers [1]. This section outlays some literature on SCA tool usability aspects, statistics on how common these issues are, and developers' thoughts on them.

Warnings

Most SCA usability issues mentioned in the literature related to how warnings are generated, selected, and presented.

One common issue is the large number of warnings that are commonly generated, making the cost of manual inspection very high [45]. In the interviews conducted by Johnson et al. [1], participants mentioned that it would be easier to deal with a large number of warnings and false positives if they were better displayed in the interface.

Another problem is the often large number of false positives among these warnings; the amount, which can be in the thousands, has been identified as one factor which hinders the adoption of SCA tools [1]. Meanwhile, a very small amount of false positives might mean that the analysis is less intricate and finds fewer true positives. Therefore, it is useful to know to which degree developers are willing to overlook false positives to gain the benefits of SCA tools. Christakis et al. [43] found that half of the participants in their study valued finding many true positives even though it came with the cost of false negatives. In contrast, the other half had the opposite opinion. The study's authors recommend that the false positive rate not fall above 15-20%

As explained in Section 4.1.2 above, another issue with warning messages is that they often are not properly understood. Bad warning messages are a complaint found in much of the research surrounding usability issues in SCA tools – developers feel that the warnings do not give sufficiently clear information regarding the problem and how they should approach fixing it [1], [2], [17], [43]. In a SCA tool evaluation by Nachtigall et al. [44], 30 out of the 46 tools were judged to have bad warning messages – only three tools were considered to give good warning messages. The messages were considered flawed based on lacking things like explanations for why something is an issue, examples for further understanding, further resources to read more about the issue, etc. Furthermore, almost none of the investigated tools had sufficiently implemented support for developers during the fixing process, such as step-by-step guides, previewing of fixes, examples, and presenting alternative fixes. In this category, a feature offering quick fixes was the most widely used, with a little over 20% of tools providing this feature.

Workflow integration

One of the most important factors for avoiding SCA tool abandonment is that the SCA tool integrates well into the developers' workflow and does not interfere with it [1]. Johnson et al. [1] found in their study that most participants were unhappy with how SCA tools integrate into their development process. Do et al. [42] found that the vast majority (74.5%) of developers prefer to have one single place of integration, where multiple tools would report

their results to the same place. It is especially important that the tool does not force developers to leave the environment they already use, especially if they are using an IDE. Christakis et al. [43] found that most developers would prefer the output of the analysis tool to be displayed in the editor; in second place came displaying in the build output, then in code review, and last in a browser. Similar results were found by Do et al. [42], which notes the two most popular places to display analysis results as the code editor and the build output. The results differ from there, where code review came in fifth place after displaying in the dedicated tool and email.

4.2 Software user feedback

Getting users to give feedback is generally a difficult task and often faces low user engagement [6], possibly due to multiple factors. This section outlines current research on how feedback can be collected, why it is difficult to reach high user engagement in feedback-leaving tasks, and possibilities to increase this user engagement. One option to increase user engagement may be to use gamification; research on gamification and its possibilities for feedback collection is further studied in Section 4.3.

Most studies in this section look specifically at feedback acquisition for software systems, and it will be noted when this is not the case.

4.2.1 Feedback types

In a mixed-method study by Almaliki et al. [3], they looked at users' preferred method for giving feedback through a survey with 100 participants and identified two different feedback types: explicit and implicit feedback.

Explicit feedback is collected by prompting the user to give feedback and consists of the subtypes: qualitative feedback, quantitative feedback, and a combination of the two. (Generally, explicit feedback was not appreciated, receiving negative responses from 70% of participants.) Leaving qualitative feedback includes writing free-text descriptions, which was not particularly popular among participants in the study by Almaliki et al. [3] (only 9% preferred it). Quantitative feedback included simple ratings and multiple-choice questions and was the second most popular option in the study (48%

Implicit feedback is when feedback is collected without explicitly prompting the user to provide feedback, for example, by gathering data from usage patterns. In the Almaliki et al. study [3], this was only preferred by less than 20% of users, which the authors explain with users' concerns about privacy and ethics. However, Stade et al. [5] find that users have different wants and needs regarding feedback providing; some are concerned with privacy while others are not.

4.2.2 Feedback acquisition methods

There are different ways user feedback can be acquired; Almaliki et al. [3] identify four different methods to gather feedback and study how popular these methods are among users; the results are presented from their 100-participant study.

The most popular feedback option, according to Almaliki et al. [3], was to give feedback online while using the software that feedback is given on, for example, via a popup directly in the program; this was appreciated by 54% of participants. This is confirmed in the study by Dzvonyar [6], where the results suggest that most, if not all, participants agree that being able to give feedback directly in the application was very useful – 87% strongly agreed with this, on a 5-point Likert scale. Another popular option according to Almaliki et al. [3] is for users to give feedback on their own accord, without being explicitly asked by the program, for example, by submitting a form online; this was appreciated by 51%. The two other feedback options presented by the same study [3] are: Being prompted for feedback, for example, via email, after using the program (appreciated by 33%), and being prompted directly in the program to go to a specific site to provide feedback (appreciated by 31%).

Regarding feedback acquisition, there is also a question about the privacy of the feedback leaver, and how the data is handled. As previously mentioned, Stade et al. [5] found that users have different opinions on the importance of privacy; some may not want their feedback to be posted online, while others want this in order to get input from other users, for example in a forum.

4.2.3 Likelihood to give feedback

Several different factors linked to users' motivation to respond to feedback requests, either increasing or decreasing the likelihood of receiving feedback, have been identified across various studies. Some of these factors are outlined in this section.

General motivations

In general, users seem to have low motivation to give feedback. In a study by Dzvonyar [6], 73% of participants said they very rarely (or even never) give feedback on applications, despite knowing the value of doing so. Their primary motivation for giving feedback at all was when they encountered a problem, or they disliked something about the application. However, these findings were abstracted from only 15 interviews about a specific feedback system called CAFE and may, therefore not be entirely generalizable.

Timing, quantity, and frequency of requests

Three important factors of feedback response rates are the timing, quantity, and frequency of feedback requests.

First, interrupting users in their current task, by requesting feedback, was found by Fotrousi et al. [46] to be perceived as disturbing to users. Similarly, Almaliki et al. [3] found that 75% of participants were less likely to respond to feedback requests if they were being interrupted in their workflow.

Second, burdening the users with too many feedback requests is likely to negatively impact their willingness to give feedback [3], [46], sometimes even impeding the usage of the software itself [3]. In the study by Almaliki et al. [3], 53% of the participants said that they often ignore feedback requests that show up too often and generally consider it spam. Furthermore, 21% said that they tend to stop using software that sends large amounts of requests, but 13% said that they do not mind the feedback requests as long as they are not forced to

answer them. Interestingly, 7% said that they were more likely to give negative feedback if they were annoyed by the requests.

According to Pagano and Maalej [47], users' general motivation to give feedback can also lessen over time. In the study, [47] found that most feedback is given during the first days after an application release, and the feedback quantity quickly decreases after that.

Perceived effort

The effort associated with giving feedback is another influencing factor for feedback-leaving motivation since users generally do want to spend a lot of time giving feedback [3]. If the effort is perceived to be high, this may decrease response rates [3], [5]. This is connected to the simplicity of the feedback-leaving interface; providing less time-consuming options, like multiple-choice as opposed to text feedback, can increase user engagement from most users [3].

If feedback is to be collected spontaneously by users, it is extra important that it is both fast and easy for the user to leave feedback [4].

Purpose and impact

Knowing the purpose of the feedback collection and the positive impact the feedback may have on the system is also a contributing factor to feedback-leaving motivation [3]; users feel more positively toward giving feedback if they feel that it serves a purpose [46]. Moreover, Stade et al. [5] found that if the feedback receivers do not respond to the feedback, users might feel that nobody is listening, which might influence them not to leave feedback.

Similarly, suppose users do not feel that the feedback request aligns with their purpose and impact wishes, i.e., it does not allow them to provide the kind of feedback they want. In that case, this may negatively affect their motivation to provide feedback [46].

Familiarity and opinion of software

Another factor influencing feedback-leaving motivation is the users' familiarity with the software they are to leave feedback on; this affected 42% of participants in the study by Almaliki et al. [3]. Furthermore, if users do not have an opinion about what they are to give feedback on due to little exposure, this might decrease their motivation [46].

Social factors

Persuading users to give feedback through social factors can be effective [48] (however, note that this study did not specifically look at feedback on software). In the study by Almaliki et al. [3], two notable social factors were mentioned. First, it did not seem important to 63% of users whether the feedback-giving was seen as a "social or game activity", for example, by being able to see what feedback was provided by other people in their circles – however, 37% (the rest) indicated positive interest. Second, being recognized in some way socially as a feedback contributor seems to be somewhat of a motivating factor (for 57% in Almaliki et al.'s study), but this can be both positive and negative, as some participants pointed out that they would prefer to be anonymous and not show their feedback to people in their social network.

Feedback request

The way in which the feedback request is designed also affects users' motivation to give feedback. The most important factor, according to [3], is how well the request design and content fits into the context where it is given, for example, being responsive to the use of smartphones and tablets and adopting how much information is given to the user. The tone and clarity of the language used in the feedback form also seem to be a contributing factor for feedback motivation [3].

Previous feedback

Almaliki et al. [3] also found that users' willingness to provide feedback increased when only a few people previously had provided feedback on the same service, although some participants said that this did not affect their motivation. Meanwhile, around half of the participants in the study were more likely to give feedback if they could see other people's feedback and compare it to their own. The rest of the participants claimed that this had no effect on their motivation.

4.2.4 Feedback response statistics

This section outlines some statistics on user feedback found in previous research, more specifically on the common types of content, level of detail, and quantity found in collected user feedback.

In a study looking at feedback reviews on applications in market stores, [47] found that the most common feedback content (75%) was praise for the app. In second place came stories about situations where the app had been helpful to the user (22%).

Regarding the level of detail users are willing to give in their feedback, a study by Broekens et al. [49] found that users typically offer more details about things they have opinions on (however, note that this study did not specifically look at feedback on software). The study in [47] found that the median length of text responses was correlated to the rating the user gave of the app; the higher the rating, the less text feedback was typically entered. The authors theorize that this is due to there not being many things that content users feel need improvement, which otherwise would be what text feedback would consist of. One takeaway from this is that options to give detailed feedback can be hidden in the interface if the user does not have a strong opinion, and maybe even if they have a positive opinion – thus simplifying the interface.

It was also noted in [6] that participants who had previously been introduced to the feedback system and gotten information about its different features, provided more feedback through these features. Note that this does not mean that the users provided more feedback overall, just that the feedback-leaving features were utilized more. It is worth noting that the sample size of these users in the study was small (15 participants), which affects the generalizability of the results.

4.2.5 Categorization of users

As a conclusion of the survey analysis by Almaliki et al. [3], four different user groups were defined by how the users approach feedback collection. Cluster one (*Feedback antagonists*) and two (*Passive and stingy people*) consist of users who generally have a negative perception of all ways of giving feedback. They do not like to be asked to provide feedback, or to be reminded when they do not. However, cluster one prefers online feedback, while cluster two prefers passive data collection feedback, with online feedback as their second choice.

Meanwhile, cluster-three users (*Privacy fanatic and generous people*) have the most positive perception of providing feedback. They like to be asked for feedback, even offline, and they want it to be explicit requests since they value privacy. They might become even more motivated to give feedback if they see that other people have given a lot of previous feedback if they can see the contents of this feedback, and if they are socially recognized for their contributions.

Cluster four (*Privacy tolerant and socially ostentatious people*) also consists of people with a generally positive view of giving feedback. However, they do not like to be asked or reminded but are motivated by many of the same social factors as cluster three – but the correlations sometimes differ. This group is instead more motivated to leave feedback if they see that *few* other people have given a lot of previous feedback if they can see the contents of this feedback (*and it is similar to their own*), and if they are socially recognized for their contributions, and if they can leave feedback as a social activity. As hinted by the cluster name, these users are not overly concerned about privacy and therefore are not opposed to implicitly gathered feedback.

4.3 Gamification

This section describes previous research on gamification, including different approaches, motivations for and against gamification, and possible outcomes. More specific research on gamification for our context (user feedback systems and software engineering) is also presented.

4.3.1 Definition

Gamification was first used in 2008 and grew in popularity and use in 2010 [50] – but since the beginning, there have been multiple different contending definitions of the term. The likely most widespread definition [51] is given by Deterding et al. [50], and specifies gamification as *"[t]he use of game design elements in non-game contexts"*.

Gamification can also be seen as a type of persuasive technology, i.e., a technology aimed to affect user behavior in certain ways without coercion or force. In gamification, this can be done by increasing users' motivation for certain actions and users' ability to make these actions by facilitating user learning and confidence [52].

Taking a step back, it is essential for gamification purposes to also define what is meant by *game*. McGonigal [53] lists four defining traits of games: they have a goal (giving a sense of purpose), rules (to motivate creativity and strategy), feedback to the user (gives motivation), and voluntary participation (making the game a safe experience).

4.3.2 Goals of gamification

Gamification has been applied for multiple different purposes. The general idea is that games can make users more focused and motivated and that game elements can create a more fun user experience for tasks often perceived as tedious and see low user engagement [54], e.g., knowledge sharing and feedback collection.

The possible outcomes of gamification can be divided into behavioral and psychological outcomes [33]. Behavioral outcomes consist of changes to users' behaviors, for example, increased interaction with certain features. Psychological outcomes are inward positive results on the users, for example, their perceived motivation and enjoyment [33], creativity [55], or overall job satisfaction [40].

Two reasons for gamification, often mentioned in the research, are motivation (psychological) and user engagement (behavioral). These are described in more detail in the following sections. In addition, gamification has often been suggested to solve social problems, e.g., to increase collaboration and team and organization awareness [55]. Gamification can also be applied to create a more coherent internal company ecosystem and community [40].

Motivation

There are many different theories of what motivates users (specifically developers) and how gamification can leverage this. Using game elements for increasing user motivation is based on the idea that games, at least good games, often include tedious and repetitive tasks but still manage to motivate their users [54].

Motivation is often divided into two categories: Extrinsic motivation is the want to do a task in order to get a reward or avoid some punishment, e.g., get praise or an increased salary, while intrinsic motivation is the want to do a task since the task itself is found enjoyable or meaningful [56]. Gamification is strongly connected with the idea of increasing the users' intrinsic motivation [57], but the methods used are often to add different types of extrinsic rewards. According to McGonigal [53], the rewards that users get out of games should not be the points or badges per se, but rather more deep-laying rewarding feelings, such as satisfying work, the experience or hope of success, and social connections [53].

In gamification research, one extensively cited motivation theory is the Self-determination theory (SDT). The theory states that three human needs must be fulfilled in order to achieve intrinsic motivation:

Autonomy is the need to have a perceived free will [57]. It regards both the user's decision freedom and how much the task seems to align with the user's own goals, i.e., if they find the task meaningful [58].

Competence is the need to feel competent and efficient [57], and the need for mastering activities and environments [59].

Relatedness is the need to feel connected to others [57], and to the community as a whole [58]; or, in another interpretation, the need to see that one's actions have consequences (in the game world) [60].

SDT has been connected to multiple gamification aspects and has served as an explanation for why certain game elements manage to motivate users.

The motivation of developers specifically has also been studied in the gamification context. One of the most important motivators for a developer, according to Foucault et al. [61], is to identify with the task at hand, i.e., to feel that the task is meaningful to them. However, this might be difficult to achieve for some more tedious tasks [61], such as feedback collection or using SCA tools.

User engagement

Increasing user engagement is a common goal for applications that struggle with motivating users to take part in activities. However, user engagement (and developer engagement) is somewhat difficult to define and is not the same as mere participation in a task, though it is sometimes measured in this way [40]. According to Stol et al. [40], developer engagement is achieved when the developer identifies with the software development community and tries to improve their own development skills.

In gamification research, the most commonly addressed user engagement problem is in systems where users should add information and content into software systems [62]. Engagement is also an issue in environments that need a set amount of people, a community, in order to be meaningful for users to participate in [63], e.g., online forums.

4.3.3 Gamification approaches

There are many different approaches to gamification; this section first defines in which dimensions gamification approaches may differ and then describes the general, common ways of designing gamification, as well as which game elements have been used or suggested in research.

Levels of game design

Gamification design efforts are usually divided into categories based on granularity, e.g., game mechanics, game elements, and game interface elements (as stated by Darejeh and Salim [62]), though the terms are often named and used slightly differently between studies, making it difficult to keep the terms apart in a systematic way. In this thesis, we will mainly concentrate on game design methods (the process of designing gamification) and game design elements, which could be categorized as high-level and low-level elements for simplicity.

It is, in general, also difficult to limit what can be counted as a game element and not [50]; i.e., which elements can be said to bring gamification to an application. Game elements have been defined as elements "characteristic" for games, i.e., commonly appearing in games, though not necessarily in all games, and not as commonly found in applications that are not games [50]; though this is not always an unambiguous definition.

General gamification approaches

According to Kapp [64], there are two types of gamification: structural gamification and content gamification. Structural gamification is the idea of adding a layer of game elements on top of an existing application while keeping the actual content of the application non-game-like. Content gamification is to make the actual content in the existing application more

gameful and fun. Of these, structural gamification is by far the most common in practice and in research, though this is not necessarily true for all gamification contexts [62].

Gamification systems often turn out to revolve around either inducement prices (competition for points, leaderboard positions, and similar), collective action (collaboration and crowdsourcing), or virtual economies [31]. In most gamification studies, the focus is on low-level game elements, such as the inclusion of badges and leaderboards. Meanwhile, multiple studies, e.g., Morschheuser et al. [32], and Do and Bodden [60], advocate for a more complete gamification, since simple game elements may not be enough to facilitate a good game experience [32]. In fact, successful gamification systems often include multiple game elements, making them more multi-faceted [32].

In general, gamification should, according to Marques et al. [65], not just be added on top of a service; instead, it should be tailor-made to solve real problems, previously noticed by the users, so that the users experience that there is real meaning for the gamification. The needed context knowledge should be collected in a large pre-study in the context where gamification is to be applied.

High-level game elements

This section lists the elements we have found in previous studies which may be categorized as high-level and considered in a gamification solution. Multiple synonyms of these elements exist throughout the research; the names used here will be the ones we reference throughout the rest of the report, regarding design decisions and similar.

Playful user interface is to use playful game aesthetics in the gamification application, for example using visual metaphors. This may be used to make a clear distinction toward other, more dull tools [40], or in order to create an emotional response in the users [54].

Social reputation is the idea to leverage users' want for a good social reputation, for example, by connecting social reputation value to elements such as leaderboard positions, ranks, or badges, and thus increasing the perceived worth of these tokens [40].

Immediate and continuous feedback keeps users informed about how they are doing in the game [62]. According to Foucault et al. [61], feedback should be presented in as small, precise, and actionable chunks as possible. Individual feedback can also promote collaboration and teamwork since it serves as a ground for discussion, comparison, and goal-setting [61]. Furthermore, positive feedback supports the competence mechanic of Self-determination theory [57].

Clear goals , both long-term and short-term, are emphasized as important by many gamification studies [56]. For example, the system can provide the user with specific tasks or milestones [62].

Teamwork can be in the form of in-game groups, chats, and social network connections. This supports the relatedness mechanic of Self-determination theory [57], [58]. Teams can induce conflict, collaboration, and competition [58]. In teams that use a gamification system together, Foucault et al. [61] suggests assigning an organizer role to one teammate, who can serve as a traction force for making others interested in the gamification tool [61].

Collaboration is when a few players or a group work toward the same goals. According to Snipes et al. [66], many developers might even be more motivated with team goals and collaboration than with individual game goals.

Competition against other users is a common game element [56], but might be difficult to get right, since it may not be appreciated by all users. It could also be common for users to prefer competition within smaller settings, such as in teams rather than on an organization level [67].

Self-expression allows the user to show their uniqueness via the game [56], for example by the use of an avatar.

Altruism is when one player's positive actions lead to rewards for a whole group of players, or the whole game world [56].

Progressive difficulty means that the difficulty increases throughout the game-play, and is matched toward the player's increased skill level. This, compared to serving the user all difficulty levels' tasks at the same time, is also good for decreasing the cognitive burden of the player [54].

Fiction/Narrative is the inclusion of scenarios, themes, stories, and/or problem settings in the game, often in the context of fantasy [54].

Framing is the act of adapting the game mechanics for the specific user, for example by personalization, by including/excluding certain elements, and by changing the difficulty based on the user. The skill level and preferences of the user can either be directly entered by the user or interpreted from user data [54].

Surprise and unexpected delight can be included to make a system more gameful and fun. An example is to give a sudden and unexpected badge for "having great hair" [63].

Collecting is the in-game act of collecting tokens such as badges, which is an activity that may motivate some users. For this to be effective, there must be some scarcity of the objects that are being collected. The collective behavior can be made more complex by allowing trading of the items [63].

Organizing and creating order might be another motivating drive for some users in a game environment, for example, if there are possibilities of arranging virtual cities [63].

Gifting may occur in the game if users are allowed to trade or transfer items with each other [63].

Low-level game elements

In this section, we list some low-level game elements extracted from previous research. According to Priyadi et al. [68], the choice of (low-level) game elements may be the most important decision for the success of a gamification system.

Points are a numerical measure of a user's progress in the game and are typically used as a reward for user involvement [69], and as a way to provide the user with individual feedback [55]. Points are supported by the competence mechanic of Self-determination theory [57], [58].

Achievements defines objectives to be reached by the user [55]; e.g., reaching a set amount of points or completing a challenge. Typically, the user knows beforehand which tasks will lead to completing an achievement, thus achievements can serve as goals for the user. The achievement also commonly leads to some type of reward, beyond the achievement itself [56].

Badges are visual representations of achievements [55], often presented in a medal-like shape [62]. Badges may act both as feedback that the user has done something good and as visual status symbols. A badge commonly aims to steer users toward certain actions [70]. Badges are supported by the competence mechanic of Self-determination theory [58].

Virtual economies is the use of some in-game currency that can purchase in-game and/or out-game goods. However, the line between virtual and real economies typically becomes blurry and is difficult not to cross [31].

Avatars are visual representations of a user's profile which may be constructed and changed by the user [55], [62]. Avatars increase the users' decision freedom [58] and are thus a motivational drive according to the autonomy mechanic of Self-determination theory [57].

Awards are similar to badges but are typically given on a more exclusive basis. An example is rolling awards, where an exclusive title such as "best player this week" is given to a player for one week [71].

Leaderboards are visual representations of competitive user rankings based on points, engagement level, or similar [55]. There are two main types: classic leaderboards, showing the high score of the most high-ranking individuals; and the no-disincentive leaderboard, where the player sees themselves in the middle and mainly gets to know how far it is to the position above and below them. It could, however, be preferable to let players customize leaderboards, or base the leaderboard view on player skill level [63]. Leaderboards are supported by the competence mechanic of Self-determination theory [57], [58].

Levels define different steps for the user to reach and can be used to demonstrate progress [55]. A user typically levels up after certain achievements, but levels are often not linear [56]. Similar game elements mentioned in some sources are *status* and *roles*, where the user is assigned in-game roles such as "Officer" to show their status in the game [62]. Levels support the competence mechanic of Self-determination theory [57].

Story is an underlying narrative of the game, which aims to put the other game elements into context and give them meaning. A story can be in the form of a scenario, or be theme-based [62]. A meaningful story plays on the Self-determination theory autonomy mechanic since it gives task meaning, and it can also be connected to relatedness if the story gives rise to a common goal in the community [58].

Quests are tasks or challenges presented to the users [55], often in a surrounding story to make them more engaging [69], and are often used to drive user action [55]. Quests are often unlocked after the user has achieved a certain level or some other task [56].

Challenges are similar to quests; they are special tasks presented to the users, often with a reward as a goal, but are more often than quests performed in competition with other users, and often under time pressure [62]. Challenges are motivated by the competence mechanic of Self-determination theory [57].

Visible progress is often shown in the form of progress bars or a percentage value, to keep users informed about how close they are to some goal [62].

Performance graphs gives the user information about their performance and thus lets the user compete against themselves [58]. Performance graphs are supported by the competence mechanic of Self-determination theory [58].

Social graphs represents a social network in the game and can be used to showcase collaboration and community [55].

Maps are visual representations of the game world and also often show the user's progress and tasks to perform [62].

Voting is when users can vote on different game options or other users' achievements. If a user gets a lot of votes, this often results in some in-game reward [69]. Voting is common for example in gamified requirements engineering, where users vote on requirements [72].

Betting is the possibility for users to bet on the outcome of certain events. The winner of the bet usually receives a reward [69].

Time-pressure and time-limits can make task execution both more engaging and more efficient [62].

As found by Priyadi et al. [68], the most commonly used game element in current research is points, followed by levels, badges, challenges, quests for social engagement, leaderboards, voting, and betting.

Reward types

Rewards can come in many different forms, such as badges, levels, and in-game status. Rewards can also be grouped into contextual categories. Some different reward types, as grouped by Darejeh and Salim [62], are: fixed action rewards, when the action that triggers the reward is clearly stated for the user; sudden rewards, when the trigger is not clearly stated; random rewards, where the trigger is stated but the nature of the reward is unknown; rolling rewards, which are given at random to someone from a certain pool of users; price pacing rewards, where multiple sub-rewards lead up to the main reward; and social rewards, which are given to the user by other users. Of these, fixed action rewards is the most common reward type.

As described by Darejeh and Salim [62], there are also three main ways to use rewards: achievement game, e.g., a badge or points as a sign of progress; in-game rewards such as the

possibility to level up or buy something with virtual currency; and out-game rewards, e.g., real money or gifts. In existing gamification solutions, the achievement game is the most commonly implemented reward usage [62].

4.3.4 Gamification in different contexts

The success of a gamification system will depend on how well-catered it is to the context where it is applied, as well as the culture and preferences of the involved users [56]. Therefore, before developing a gamification service there should be an understanding of both the underlying context and the users' needs, motivations, and behaviors [32]. In this section, the impact on gamification of context, culture, and user types is examined.

Context

Gamification seems more fitting for some application areas than others. The best contexts, according to Hamari et al. [33], are those in which users continuously use the service (and not only sporadically), in order for them to have time to get invested. Currently, the overall most common gamification context is education and learning; here, gamification has also been extensively shown to have positive outcomes, though with some possible caveat from the increased competition [33].

Culture

The culture (and company culture, for industry gamification) is important to consider in the gamification design [56]. Cultural biases significantly impact how users react to elements that try to change their behavior, and which behaviors feel most natural for them in the first place. These cultural biases can often be seen in a person's or a group's habits and decision-making. For example, does the culture mostly value individualism or collectivism; do people view themselves as dependent on a group or as independent individuals? Because of this, it is important to study the culture of the intended users before finalizing any gamification design.

User types

The users' qualities, demographics, and expectations also heavily affect the success of a gamification effort [73]; different users interact with game features for different reasons, in different ways, and with different results [74]. Generally, it is not possible to use a one-size-fits-all approach to gamification, so the service should be developed to fulfill different types of users' wants and needs [69], [74]. The game design should preferably also be customizable to allow for users' self-expression, though the user should not be overloaded with the need to make a lot of decisions [63].

Examples of demographic factors that can affect users' interaction with gamification elements are age, sex, and previous gaming experience [73]. A lot of research has been done on the social dynamics of regular games, and according to Barik et al. [54], this research should also be considered for gamification efforts. For example, some studies show that female players, compared to male players, are often more interested in social game aspects than competitive

aspects. Such findings motivate the usage of more narrative and social game elements, as a complement to competitive elements, to appeal to a more diverse user base [54].

Furthermore, there are differences between users' individual preferences. Humans enjoy playing games for many different reasons, which are likely to differ between persons. For example, some enjoy the game mechanics of collecting and ordering things (like building nice, orderly cities), while others enjoy the status and recognition they get from achievements [63]. Some users might also especially dislike some game elements, such as competition features [33]. Categorizing players is a large subject within regular game design. However, these frameworks were often designed for MUD games such as World of Warcraft, and even though they are commonly used for gamification design, this is not a good practice according to [35].

An alternative player categorization framework, created especially for gamification, is the Hexad types [35]. This framework is based on motivation research, for example, Self-Determination theory and the idea of intrinsic and extrinsic motivation, which might motivate different users to different extents [35]. The accuracy of the framework has been empirically evaluated with positive results in a follow-up study by Tondello et al. [37]. The list below goes through the six Hexad player types and suggested game elements for each type, as described by Tondello et al. [35], with improvement suggestions from the follow-up study [37]. The suggested game elements are sometimes slightly tweaked by using synonyms or similar, in order to map directly to the list of game elements given in Section 4.3.3 and Section 4.3.3 (these elements are italicized for clarity). Some elements referenced in the Hexad framework which do not map to our previously listed elements are also mentioned here.

Philanthropists are motivated by helping others in an altruistic way, without wanting any extrinsic reward, and by finding a more important purpose and meaning in the game. Suggested game elements: *collecting*, with trading possibilities; and *gifting*. Also knowledge sharing and the use of "administrative roles".

Socializers are motivated by the relatedness mechanic of Self-Determination theory. Their main purpose in the game is to socialize and interact with other players. Suggested game elements: *teamwork*; and (social) *competition*. Also social comparison and "social discovery".

Free spirits are motivated by the Self-Determination theory's autonomy mechanic. They value freedom, self-expression, and exploration, and do not like being told what to do. Suggested game elements: (exploration) *quests*, *surprise and unexpected delight* (in the form of "Easter eggs"), *self-expression* (in the form of creativity and customization), progressive content-unlocking; and, according to the follow-up by Tondello et al. [37], *challenges*, learning, anonymity, and "anarchic gameplay".

Achievers are motivated by mastery and self-improvement, i.e., by the competence mechanic of Self-Determination theory. They enjoy tasks, progressive difficulty, and proving themselves. Suggested game elements: *challenges*; (epic) *challenges*; *quests*; progressive *levels*, and "learning new skills". However, according to the follow-up by Tondello et al. [37], achievers are not motivated by epic challenges but are slightly motivated by *badges*, *achievements*, and anonymity.

Players are motivated by the actual, extrinsic game rewards, such as the badges, and value these more than the activities leading up to the rewards. Suggested game elements: *points*, *leaderboards*, *badges*, *achievements*, *virtual economies*, lotteries/luck games, and rewards in general. Though, according to the follow-up by Tondello et al. [37], players are not motivated by lotteries or luck, but instead by *levels*, *collecting* and trading, social *competition*, *challenges*, *quests*, progression, social comparison and discovery, and anonymity.

Disruptors are motivated by change. They want to sabotage the game and interfere with it, by positive or negative change (disturbing or improving the system) or even by cheating. Suggested game elements: *voting*, anonymity, "innovation platforms", and "anarchic gameplay". Though according to Tondello et al. [37], disruptors are not motivated by anonymity; instead, they are motivated by *challenges*, social *competition*, and creativity.

Note that there can be some overlap between the different Hexad types and that most players will have traits of multiple types to some extent [35]. It may also be common that players take on different roles depending on the game, and may also alternate between types or adhere to multiple types at the same time [56]. In the empirical evaluation of Hexad [37], it was found that the most common types are Philanthropists, Free spirits, and Achievers, and the least common type is Disruptors. It was also found that gender and age correlate to the user types [37].

4.3.5 Efficacy of gamification

It is generally difficult to evaluate if gamification as a concept has the ability to achieve the goals of increased user motivation and user engagement (or other stated goals). This is partly due to the term having been popularized and often evokes strong, differing opinions among users and researchers, sometimes mostly based on anecdotal evidence [33], [69]. There are also difficulties with evaluating different game elements in isolation, and the endless variation of gamification designs and contexts in research lead to very different results [33]. As noted by Sailer et al. [58] and Morschheuser et al. [32], reported effects on gamification might be misleading since many game elements are often evaluated together; especially successful gamification solutions often include multiple game elements [32].

Secondary studies of gamification experiments showcase that almost only positive results of gamification have been reported in research. In the 2014 literature review by Hamari et al. [33], gamification is concluded to "work" for its purposes, although with some caveats; all the found studies reported positive effects of gamification at least from some users, but most studies also reported some neutral or negative effects from other users. Similarly, in the 2016 literature review by Darejeh and Salim [62], which focused on gamification for user engagement, 80% of the studies reported positive results, 0% negative results, 8% neutral results, and 11% partially positive results. The authors conclude that gamification is suitable for increasing user engagement, and also that gamification does not have any serious impediments to the usability of software. Also, the 2015 literature study in Seaborn and Fels [73] noted that gamification research gives mainly positive results, though inconclusive.

However, there may be many reasons for these positive results: They might be because of the relatively small number of studies, which perhaps cannot give a reliable mean value of

gamification effectiveness, or the "file-drawer effect", i.e., that only positive results get published [73]. The positive results might overshadow legitimate concerns about gamification risks and dangers, which have also been indicated by a small number of studies [73]. Another explanation of the positive results could be the novelty effect of the new gamification service, which may give behavioral short-term user outcomes (which is what is typically measured in these studies) but no long-term effects [33].

In a large experiment (N=419) on the effect of different game elements, Sailer et al. [58] concluded that gamification might not be effective per se, but that different game elements can impact user motivation in different ways. For example, the users' feeling of competence and autonomy was affected by the change in task meaningfulness introduced by badges, leaderboards, and performance graphs. Meanwhile, relatedness was positively influenced by avatars, meaningful stories, and the existence of teammates.

4.3.6 Possible pitfalls

While gamification in general has shown positive results, it has faced a lot of criticism, and there are many challenges for gamification designers to overcome. Some of these possible pitfalls, commonly raised by studies on the subject, are described in this section.

Bad game design

In 2012, Gartner [75] predicted that 80% of the gamified applications in 2014 would fail to meet their business goals specifically because of poor design, a prediction that has been cited by many gamification studies since. Gartner argued that most gamification efforts were too driven by hype and too poorly designed to succeed; too focused on merely adding simple game elements such as points and badges rather than on more complex issues such as balancing rewards and the usage of virtual currencies.

In many ways, game design is a challenging matter. Since gamification must also work together with already existing systems, for example, a software tool stack, there is also the major challenge of integrating the gamification aspects into the existing services [69]. There is currently a lack of research on how such integrations should optimally be done, and most research experiments with gamification are ad-hoc solutions; at the same time, gamification might only succeed if it builds on a smooth integration and is not just a new, standalone tool [69]. This is especially important in software engineering, where the tool stack infrastructure often is part of the organization's culture and a key part of developers' workflow [69]. One commonly brought-up problem with gamification in software engineering is that it may disrupt the workflow and become a nuisance for developers, thus it is important to design game elements so that they do not interfere with effectiveness and productivity [60].

It is also important to avoid bugs in the game design; otherwise, developers might change their habits to get around the bugs. An example is found in the study by Foucault et al. [61]: There was a bug in the gamification score counter for merges, making some developers try to merge their code less in order to not lose legitimate points.

Shallow gamification

Another common criticism of existing gamification solutions, and gamification in research, is that they are so-called *shallow* solutions [76], or *narrow* as called in Barik et al. [54]. Shallow gamification only focuses on one dimension of game design, namely the reward system, often with fixed rewards in the form of points, badges, and leaderboards as extrinsic motivations [54]; this is overall the most common gamification approach [62]. Often, these designs are directly or indirectly based on the idea of behaviorism, i.e., that rewarding and punishing behaviors will steer later long-term behavior patterns - but behaviorism has been thoroughly proven not to work long-term since users' behaviors only change while the reward or punishment is present [31]. Partly because of this, shallow gamification has been criticized by many experts in the field [76].

Another problem with reward-focused shallow gamification is that user's pleasure will not increase with the number of rewards, or, as phrased by Groh [59]: "Pleasure is not additive". This type of gamification might even have harmful long-term effects, as the experienced pleasure from an activity may decrease after the removal of gamification features, to a level lower than initially [59]. Reward- and competition-based gamification may also scare away some users while appealing to others, which impedes diversity and can give long-term negative effects on the platform [54].

Gamification systems could, according to Darejeh and Salim [62], likely reach higher quality if they included more game mechanics and elements. As said by Barik et al. [54], a better gamification solution would be to focus more on the complete system and re-imagining the system as some type of full-fledged game. At least, the elements included in the system should align with the actual purpose of the tasks the users are carrying out [54]. Another way to make gamification less shallow is to add framing; user personalization such as avatars; narrative; and fiction, for example by adding small, playful narrative aspects to badges [54]. Overall, a mixture of various game elements and reward types should be used, e.g., both in-game and out-game rewards, as recommended by Darejeh and Salim [62].

Encouraging the wrong behavior

Adding an extrinsic reward system also comes with the risk of shifting the users' focus toward the gamification features and away from the actual, meaningful task that the system was originally intended for. Extrinsic rewards may come to substitute the intrinsic motivation for the task, making the task seem even less fun than before, and change developer behaviors to focus on getting game rewards instead of optimizing their work [33]. For example, if rewards are given mainly for quantitative work, users might change their behaviors to do smaller, less meaningful, or unnecessarily fine-grained tasks, as was noted for example in the study in Frącz and Dajda [77]. In a study on user behavior in the gamified StackOverflow forum [78], it was discovered that users would work hard with certain tasks before acquiring a badge, and then rapidly decrease this activity when the badge was acquired. This implies that user behavior shifted toward the game goals instead of the actual task goals. This shows that the badge "works as intended", but this type of user behavior might not always be a desired outcome of the system. The phenomenon also goes against one of the main ideas of gamification, to increase intrinsic motivation. Replacing real incentives with fictional incentives has been referred to as "exploitationware", and it may be especially bad if the original task was, before gamification, perceived as rather engaging [31].

Another risk with an extrinsic reward system is that it may be perceived as lessening the players' autonomy (which according to SDT is one of the three main intrinsic motivators). The users might feel micromanaged and the reward system may seem to coerce them to do actions they did not choose themselves. This can be mitigated by using more informational game feedback than controlling feedback, and by having shared goals on the platform rather than individual goals, even if they are pursued in an individual fashion by users [59].

Related to unwanted user behaviors, cheating and gaming the system are common problems for gamification systems, especially regarding gamification for software developers and other software-confident users [79]. These issues must be addressed by using a well-designed and fair reward system in order for the gamification solution to remain enjoyable and fair for everybody.

Not appreciated by all users

Some people, for example, some developers, hold strong negative opinions on gamification or aspects of gamification. Some reasons for this (brought up in developer interviews in the paper by Snipes et al. [66]) are that gamification is seen as childish, can increase competition between teams in an already competitive environment, and that developers should be motivated by doing a good job rather than by game rewards.

The fact that gamification does not attract all people highlights the importance of keeping participation voluntary. This can be achieved by making the game elements opt-in, so that developers can choose when and how to participate, and optimally also with whom they share their data [61]. This idea of voluntariness also goes hand in hand with viewing gamification as a persuasive technology, which does not force users with coercion but rather only influences them to change their behaviors [52].

Perhaps most importantly, gamification solutions should only be applied when trying to solve real problems noticed by the intended users [65]. Otherwise, users might not understand the reason for gamification and perceive that game elements are not there for their sake but for some other reason, which can lead to frustration and unwillingness to participate in the game elements.

Privacy and data control

One commonly raised worry about gamification in industry settings is the vast amount of user data that is commonly collected and displayed [31]. Users may experience the data collection as a mechanism for management surveillance, evaluation, and control, which can lead to worries or repulsion for the game elements [31], [67]. This can especially be an issue when there are leaderboards that might serve as simplified, and sometimes misleading, charts over employee performance [61].

This issue can be solved by increasing the anonymity of the gamification tool [67], and/or having clear and formally stated data usage policies, both to avoid worries and to hinder actual misuses of the data [61]. Handling such ethical issues might be crucial for the success of the gamification system [61]. Also, developers are likely more positive to data sharing if it is done in a smaller environment, such as within the team, thus team collaboration and/or competitions could be used as alternative game elements [66].

4.3.7 Static analysis feedback gamification

As described earlier, the success and outcomes of gamification are often very context-dependent. In this thesis, the gamification context is feedback collection on SCA tools, in a software development environment where the users are developers. In this section, we focus on gamification of software engineering (SE) tasks, as well as feedback collection, in order to get a more specific view of current research on the relevant area.

Software engineering gamification

Gamification has been applied to many different aspects of the SE process: implementation, testing, configuration management, requirements engineering, and more [69]. In general, SE has been a popular area for gamification, and according to Pedreira et al. [69], it is a promising field for gamification because of the human-intensive nature of the processes [69]. Especially some aspects of SE, namely development, testing, and requirement engineering, share features that make them well-suited for gamification: They are often difficult, tedious, and require intense collaboration between people, commonly leading to a lack of user engagement and intrinsic motivation [69]. Another gamification-suited aspect is quality improvement of different kinds, which is usually not perceived as the most engaging task by developers [61].

In a literary review on SE gamification [51], examining 130 studies between 1987 to 2020, it was noted that the practical results of gamification are still unclear, which is similar results to those of literature reviews on general gamification.

Static analysis gamification

There is not much previous research done on the gamification of SCA tools [60]. Still, according to Do and Bodden [60], SCA tools are a promising area for gamification because of the workflow similarities with the flow of video games, with repetitive and challenging tasks, requiring a high understanding of the task and searching for solutions to problems. While video games, however, are developed for high user engagement, SCA tools are notorious for their lack of this [60], as described in Section 4.1.3. SCA tools have many features that impede users' intrinsic motivation according to Self-determination theory. For example, the slow feedback that results from the time it takes to run the analysis is directly contrary to game design recommendations and can decrease both the feeling of competence and relatedness among developers. SCA tools can also be said to provide the user with too much autonomy, since the warnings are often shown in a redundant amount and without proper prioritization, leaving this task to the user [60].

Feedback collection gamification

A few studies have been done on gamification of feedback systems, and similarly in other information crowdsourcing or knowledge-transfer settings, such as for building up organization-internal wikis [71]. Typically, tasks such as user feedback see low engagement among potential feedback-givers [80]; in companies, employees often prefer spending their time with more pressing (and fun) tasks than knowledge-sharing [71]. Therefore, this seems like a promising area for gamification.

However, different people have different motivations for leaving feedback, and these motivations can also vary considerably between cultures [80]. For example, users can be affected by the visibility, amount, and similarity of previously given feedback; some are motivated by the social recognition they may get by providing feedback, others are not motivated by this, and yet others are only motivated by social recognition if it is connected to having their feedback actually lead to visible changes. These different motivations can be utilized in a gamification setting, where social recognition can be nurtured via badges or similar status symbols, and by making previous feedback visible to users [80].

4.3.8 Gamification frameworks

There have been many frameworks developed for the design and design process of gamification systems [31], and the utilized framework can be crucial for the success or failure of gamification [30]. Most existing frameworks focus on psychology rather than technology, for example, on social influence and personalization, which might be reasonable since it has been theorized that gamification is only 25% technology and 75% psychology [30].

However, while the interest to develop conceptual gamification frameworks has grown rapidly since 2012, the presented frameworks have some shortcomings, for example in their lack of addressing ethics, balancing, and sustainability [81]. Ethical aspects are very important for gamification solutions, because of the data handling privacy needed, but also because the gamification aim is to directly impact the user's motivation and/or behavior; frameworks must effectively make sure that gamification is not implemented solely for business purposes that ignore the user's wants and well-being [81].

Design frameworks

Some gamification design processes have been specified in previous research, for example, the one by Nicholson [82], and by Dal Sasso et al. [31]. However, while some frameworks are very general (as Nicholson's), others are very specific, such as the one by Dal Sasso et al., which seems best adaptable to a badge system. This may make the frameworks difficult to effectively apply in specific contexts. However, there are also many general ideas and recommendations on how to design gamification, which may be easier to adapt.

One such general design recommendation, by Barik et al. [54], is that the game should be easy to understand and lightweight enough to not require a manual. Features should preferably be found by the user by experimentation, and if meta-information must be given, it should be in an in-game manner [54]. Specifically, the first few minutes the user interacts with the system, which are crucial for further engagement, they should not be hampered by information [63].

Process frameworks

Many gamification process frameworks of different types are presented in the 2015 literary review of Mora et al. [30]; multiple approaches are very similar to each other. Some common aspects are the importance put on business objectives (some frameworks consider economic issues and stakeholder participation in the design, others do not); and most use a human-centered design with the player and player psychology in focus, compared to, for

example, technology-based or goal-based design [30]. However, a generic, complete gamification framework does not yet exist.

Others have also highlighted the importance of catering the gamification design to the specific context and the individuals that will use the gamification service, for example by defining user personas and basing the design on these [40].

In a 2023 SE gamification literature review [68], it was noted that no studies from the last five years used a common gamification process framework and that the designers instead built the frameworks by themselves. This might imply that there, even though there are many frameworks, are no great and well-known frameworks to use, or simply that no framework has gained more traction than others.

Evaluation frameworks

There have also been some frameworks developed for how to evaluate gamification results in order to more systematically define the success of different gamification studies and be able to compare them. However, a literature review of 100 studies (2011-2020) on SE gamification [34] found no good, general framework for gamification evaluation - all studies use different approaches. There are, however, some patterns in the evaluations; usually, they focus on the user experience of the gamification elements and on the outcomes of the users and the context. The most commonly used evaluation criteria are engagement, performance, satisfaction, and motivation [34]. In the literature review conducted by Hamari et al. [33], it was noted that most studies measure behavioral outcomes.

The optimal gamification evaluation method will depend on what is to be tested, but according to Monteiro et al. [34], the evaluation should generally use a mix of qualitative and quantitative data collection and analysis, with both subjective and objective inputs. According to Hamari et al. [33], there should also always be control groups when doing gamification user tests, which historically (at least in 2015 when Hamari's paper was written) has been lacking in gamification evaluation research.

4.3.9 Current state of research

There have been many primary studies and experiments on gamification. In 2021, Porto et al. [72] noted that studies on SE gamification have increased in the last years, indicating increased awareness of the topic [72]. Furthermore, there are multiple secondary studies and literary reviews on gamification research, for example [30], [33], [73], and more specifically on gamification in software engineering, such as [51], [68], [69], [72].

However, in all or almost all studied literature reviews, there have been voiced complaints that gamification research, in general, is not of high enough quality and lacks certain types of studies. The studies conducted are said to be too basic and only include very simple gamification elements such as points and badges [62], [69], [72] - i.e., so-called shallow gamification. Also, Pedreira et al. [69] noted in 2015 that SE gamification research was still in an initial state, with most studies being workshops or conference studies and not peer-reviewed journal articles. In general, the studies seldom give any empirically valid support for gamification [69], and most studies merely give suggestions for gamification approaches without any real experimental evaluation [69]. Most studies at this time were also not systematic and lacked a sound methodological approach; for example, they did not compare a system in both

its gamified and non-gamified version [69]. Still, in 2018, Morschheuser et al. [32] claimed that research on developing successful gamification was still starting. Now, the actual empirical evidence of the benefits of software engineering gamification is limited [72].

A problem for these research endeavors is that gamification applications often are very context-specific since they commonly build on other tools, and it is thus difficult to show any empirical result applicable to general contexts [69].

Chapter 5
User research results

This section presents the results from user research conducted to answer RQ3 (*From a developer's perspective, which aspects hinder interest and engagement with static analysis tools and with giving feedback on these tools?*). First, we present a thematic analysis of the interview findings, then the questionnaire results. To conclude the chapter, a summary and conclusions of the collective user research are given.

5.1 Interview findings

For analyzing the interview data, we used a thematic analysis approach as described in Section 3.2.3. This resulted in a set of themes (T1-T4) and subthemes presented in a thematic map in Figure 5.1. In this section, each theme is described one by one, divided into subtheme sections when subthemes do not overlap too much. Note, however, that the overview of the Axis tool setup (the main part of subtheme T1.1) is not included in this section and instead serves as a basis for Chapter 2.

Since all eight interviewees use he/him pronouns, these will for convenience be used throughout this section when referring to interview participants. Participants will also be referred to as their interview ID given in Table 3.1.

5.1.1 T1.2: Views and experiences of SCA tools

The T1.2 theme presents developers' thoughts, opinions, and motivations regarding SCA processes at Axis. Some insights are also given on how the tools are utilized in practice, not only in theory as described in Chapter 2.

For convenience, since the T1.2 theme has many overlapping subthemes, the findings are presented in smaller chunks in which multiple subthemes may be handled together. Some subthemes may be split into different sections.

Figure 5.1: Resulting themes and subthemes from the thematic analysis of the interview data. Each circle with a theme ID (T1-T4) represents a theme, other circles represent subthemes.

Why SCA tools are used: motivation and coercion

Why are SCA tools used at Axis? All interviewees agree that SCA tools are, at least in theory, useful for writing better and more secure code.

Most participants express a feeling that SCA tools prevent developers from making mistakes, by catching problems in the code that otherwise could lead to serious bugs or vulnerabilities. In this way, SCA tools provide a feeling of security for developers. As phrased by *I8*: *"[SCA tools], especially static type checking, are a good complement to tests, to convince me that the code I'm writing works as I think it should work."* Even participants that are more hesitant to SCA tools, such as *I5*, mean that the tools *in theory* are very useful - if they actually flag real problems in the code.

Interviewees also use SCA tools for speeding up the development process; letting the tools catch small mistakes allows the developer to not have all details in their head and instead concentrate on moving forward. As noted by some participants, most encountered SCA warnings are about common, small problems that often are pretty obvious but may be overlooked by developers while writing the code. SCA tools may also speed up the development process by discovering mistakes and problems earlier in the process. Running the code through formatting tools and simple checks also lets developers avoid tedious discussions about these small issues during code reviews.

Another motivator for using SCA tools is that they can make the code more consistent among developers, by forcing a specific way of writing code - this is especially mentioned

by participants in regard to formatting tools. SCA tools can thus increase code quality and make it more pleasant to read; *"so that it doesn't become a patchwork of the code"* (*I1*).

SCA tools may be especially appreciated where the developers are not as used to the code base or the programming practices they are currently working on, as mentioned by one participant whose team is new to the programming language they use. He appreciates that the tools can tell him which coding practices are good and bad; without the tool, the team might have continued using bad practices. In this way, the SCA tool may serve as an "educator" and make developers change their coding habits. As said by *I6*: *"After having used this for quite many years, we have learned to program in a way so that cppcheck and Sparse, they don't find that much. It's second nature to do it correctly. However, Coverity still finds pretty much."*

Notably, different SCA tools are used for different purposes and are often perceived to complement each other. For example, one participant mentions that clangd gives good advice on best practices for coding, but cannot find logical errors, as Coverity can.

However, motivation for using SCA tools does not only stem from the developers themselves but also from company regulations and company culture. *I6*, who has worked at Axis for 25 years, says: *"Axis has also generally acquired an increased security mindset"*; this has made the company more actively, over the years, encourage developers to use SCA. However, many participants claim that they would be motivated to use the tools even if they were not enforced by the setup, though *I8* says that he thinks some developers would ignore the tools if they could.

SCA tool problems

Some participants are not at all motivated to use static analysis because of factors they dislike about the tools, and even the more motivated participants often note the same problems. The SCA problems that were most often brought up during the interviews are listed below:

Long runtime of the analysis tools is mentioned by two participants as frustrating. *"[...] sometimes it takes a long time, and half an hour later one gets to know that something didn't work, and then one has to context switch back to that change and fix it, and often it's a pretty small thing [to fix]"* (*I8*).

Slow UI is brought up by the participants specifically regarding the Coverity UI. The slow interface is a reason for developers to instead access the analysis through the HTML file.

Configuration difficulties i.e., the difficulties and effort needed to integrate the SCA tools in the development setup, is commonly brought up as a source of irritation.

False positives is a SCA-tool problem that the interviewees were specifically asked for their opinion on. Experiences and opinions varied significantly; while some barely have encountered this issue, others mention false positives as a major nuisance that may lead developers to ignore SCA warnings and even entire SCA tools.

Regarding false positives, the vastly different experiences among developers are worth analyzing. Naturally, the amount and types of false positives depend on the tool. For example, *I6* says that the false positives in Coverity usually are similar to each other and can be understood, while: *"With some [false positives] in cppcheck, it's just wrong. It didn't think. So it's worse.*

One doesn't get that many, but the ones one gets can in practice be impossible to correct" (*I6*). The nature of the false positive may also affect how many false positives developers are prepared to accept. When *I3* points out that his team previously used to have a problem with many false positives in Coverity, "many" in this case meant one to two false positives per month.

Specifically regarding Coverity, the amount and type of false positives seem to vary greatly between code bases, and participants in different teams have very different experiences and opinions. In *I2*'s team, Coverity commonly produces false positives regarding code in external libraries that are used in many places in the code base. This leads to *I2* thinking that Coverity is less suited for his team, and it affects how the team regards the output of Coverity. If *I2* gets a -1 Gerrit vote from Coverity, he says he checks it *"sometimes"*.

In connection to this, *I6* notes an interesting fact about the history of Coverity: *"In my team, we are a bit fortunate, because Coverity is written from the beginning for being run at the Linux core, that's where it comes from. So it's very good at analyzing what we use it for. Perhaps further up in the user space they have a lot more false positives, we have seen."*

SCA work process: Getting around problems

In Chapter 2, the general SCA tool setup and processes at Axis are described. However, how the tools are used in practice may differ, for example, when developers do not want to use the tools and find workarounds.

Some participants say they usually ignore some SCA warnings, either because they believe the warnings to be false positives, or just do not find them useful. Participants commonly mention Pylint and Coverity as having many false positives that they become inclined to ignore. Three of the interviewees barely use the available SCA tools at all. While Coverity is automatically run via Gerrit for their repositories, they have altered the setup so that -1 votes from Coverity can be ignored. As said by one of these participants, *I2*: *"Coverity is not a super important part of the workflow, it is more like another input that one may look at." I1* thinks that most developers in his team disregard Coverity: *"When it was introduced, people were a bit against it, then I think it slowly died out how much people cared about it."*

Sometimes, developers do not ignore warnings they disagree with and instead change their development habits in order to *"make the SCA tools happy,"* for example, by annotating their code. This is seen as a tedious and irritating factor of SCA tool use.

Understanding SCA warnings

Most participants say they seldom have problems with understanding SCA warnings; they often find the warnings clear and as providing enough information, though this is of course tool dependent. When *I8* was prompted to make an estimation for all SCA tools he uses, he says that around 1 out of 10 warnings may be difficult to understand immediately. *I3* says that he mainly has to search for more information about a warning, in a time-consuming way, when the warning is a false positive. Regarding Coverity, all participants with experience with this tool think that Coverity warnings are usually presented in a clear and easy-to-understand manner. *I1* describes the warnings as *"pedagogical"*.

Multiple participants point out that SCA warnings get easier to understand with time and that there are no problems once one is experienced. This is especially the case since the tools often flag similar problems over and over.

Handling false positives

The participants were also asked how they handle false positives, in order to analyze the problems that false positives may give rise to. From the interviews, three main options for handling false positives are most often brought up: turning off the warning; leaving the warning in place but ignoring it; and changing the code that triggered the warning just to make the tool stop flagging it.

If warnings can be turned off, and to what extent, is tool dependent; all participants were familiar with the possibility to turn off warnings in at least some tools. When it comes to Coverity, warnings can be turned off via the Coverity UI (which only developers with licenses can access), and the knowledge and use of this feature vary between participants - some did not know of it, while some use it frequently. There may also be some risks with turning off warnings, as noted by *I4*, whose team (to his knowledge) does not use this feature in SCA tools: *"Often, something that is perhaps bad at one occasion may be right at another occasion, so perhaps one does not want to turn it off for the whole code base."* Sometimes, it may also be difficult to know if a warning is a false positive or just a difficult-to-understand true positive. There is a worry among some developers that they could risk turning off warnings that might have flagged an actual problem. *I5* also notes that turning off many warnings takes away the usefulness of the tool; he thinks that in that case, it would be better to just stop using the tool.

When warnings cannot be turned off or ignored, the developer usually has to make a workaround in the code. This is how *I4* has to solve Coverity false positives (though he has only encountered a single such warning before) since it to his knowledge is not possible to override the -1 vote Gerrit in his team. *"I don't really know who I would contact … There is surely someone one could contact if one really wants to, but it's a little too much effort compared to just rewriting my code"* (*I4*).

I1's team, who could not on their own turn off warnings in Coverity, instead used to add the false positive problems as a ticket in Jira, and ask someone else to mark it as a false positive. *"And it turned out, in the end, that people just kept adding comments on this ticket, when they found false positives. So, from our perspective, we did not see that it got fixed, we just saw that we had a problem that was repeated many times"* (*I1*).

Return on investment of SCA

The participants' views on SCA seem to boil down to the return on investment of the tools: The SCA setup clearly has both benefits and problems, the question is if the benefits are big enough to endure the problems.

One important aspect is how good the tools are at finding relevant problems in the code. Different interviewees have different opinions on this. For example, *I3* says that Coverity cannot find all memory leaks; he himself has found memory leaks that Coverity failed to flag. Meanwhile, *I6* says that Coverity has great functionality when it comes to finding issues.

There is also the aspect of how often one gets warnings; if it is too often it might be irritating, but if too seldom the tool might seem ineffective and not worth using. Regarding Coverity, *I2* says that when they started using the tool they did not discover many new issues; meanwhile, *I6* says that Coverity gives the largest amount of warnings, and the most useful warnings, out of his team's SCA tools.

The usefulness of the tools can then be put against the effort of configuration and continuous use. Multiple participants discuss the cost of the tool, as in the work effort needed; time to research how to integrate it, long runtimes, and the effort to triage false positives, are examples of this. *I3* also mentions that Coverity costs a lot of money for Axis and that the tool might not be worth that money.

The verdict - if developers find their SCA tools worth using or not - differs between participants. Some definitely find the tools worth the irritation and effort. Meanwhile, *I5* concludes his dislike for SCA tools with this: *"I feel like they are not worth the work, like, since they mostly complain about things that don't matter that much. One has to put a lot of time into, like, massaging them to not complain about little things. They won't find any larger, more difficult problems anyways."* As an in-between, *I2* thinks that no SCA tool can work as a *"silver bullet"* that finds all problems in the code. Instead, he would have preferred to have a set of different tools, which might change over time, to make overall checks of the whole code base to and from. He also says that using the tools more seldom, not as a part of the development workflow, would decrease the workload.

Initiatives for changing SCA practices

Since not all developers are happy with the current SCA setup, or at least have opinions on it, it is interesting to see what developers do in order to change the situation.

The commitment to improve SCA practices vary notably among interviewees, where the participants that are generally more positive about SCA are also more engaged in changing SCA practices and finding new SCA tools. For example, both *I7* and *I8* have introduced and encouraged the use of new SCA tools within their team before. Meanwhile, the participants with an overall negative attitude against SCA seem much less willing to research SCA tools and to find other practices; to the participants' knowledge, finding better SCA tools has not been discussed in their teams. This might indicate both low motivation and a feeling of hopelessness toward SCA. A notable exception, however, is *I6*'s team: They previously used a SCA tool (Lint) which led to too many false positives and a high workload; they did not only get rid of Lint, but also created a better repertoire of SCA tools.

There is also a middle ground: *I1* expresses a wish that SCA tools were used more in his team. About Coverity, he notes: *"But for being able to use it more, one must get others to want to use it. For if one forces someone to use it and they don't like it, then I think it would be bad"* (*I1*).

5.1.2 T2: Leaving feedback on SCA tool warnings

This section goes through theme T2, about developers' thoughts on, and previous experiences with, leaving feedback on SCA tools. Some participants seem positive to the idea of leaving feedback on SCA warnings, for example, by noting if a warning was useful or not. However, multiple factors weigh in on whether participants would appreciate a feedback system or not.

One aspect is how clearly and immediately the feedback has a positive effect on the system, either so that the feedback-giver gets the benefit of a better tool, or at least some other users profits from this. *"If one has a suspicion that it [the feedback] only goes into a large database and no one ever looks at it, it feels a bit thankless to send in that type of feedback. But if it actually helps oneself or the reality gets better, then it's more interesting"* (*I8*). For example, *I7* notes that

giving feedback on SCA warnings would be useful if the tool adapted immediately and in the long term based on which warnings the user finds useful and not.

Similarly, *I1* would be positive to a feedback system if it could show which warnings had previously (also by others) been noted as false positives; this is another way in which the collected feedback could assist users. *I3* also mentions that he would want to have more SCA statistics, both in general and on repository-specific problems. *"Now it is pretty closed, one just gets to know 'here's the error', and one doesn't get to know anything more"* (*I3*).

In the same spirit, *I2* suggests that it would be good to have something similar to StackOverflow internally at the company, i.e., some sort of database with discussions and fix suggestions for SCA warnings, perhaps also including finished modules of code; this would help developers to quickly find solutions to problems, including SCA problems. However, there were also some cautionary notes raised about displaying feedback such as SCA warning categorizations. *I1*, who still thinks this would be useful, to a degree, also notes that one may put too much trust in previous feedback: *"If someone has reported it as a false positive, then one thinks 'alright, alright, it's a false positive', and then one ignores it." I2* also notes that on forums such as StackOverflow, not all information is correct, and one may question if it is the right type of people that answers the questions; if having a system that displays SCA fix suggestions and similar, one must think about the possibility that incorrect information is spread.

Meanwhile, *I4* thinks it would be useful for a feedback system to mark warnings as not helpful or as false positives, even if the developer might not be right in their assessment. That way, the developer could then get answers back from other developers that it was wrong, the thing could be discussed, and the developer would get to know what is correct instead of remaining unsure.

Another important factor for the interviewees' attitudes toward feedback systems is how much work it would be for them to leave feedback; multiple participants mention that giving feedback should not be too time-consuming. *I5* thinks he would not like a feedback-leaving system, since he already thinks that SCA tools are mostly in the way – leaving feedback on warnings would just take even more of his time.

Notably, the interviewees that are less positive about SCA tools are also less positive about leaving feedback on the system. *I2* says that he would be positive to a feedback system if it could solve the problems that his team has with a large number of false positives, but since previous complaints have been raised about this without any action being taken, he does not seem very hopeful that this will be the case. Overall, the participants, especially the ones that dislike Coverity but still have the tool integrated into Gerrit, seem to feel that they are not being listened to. This might be the reason for the common suspicion that feedback will be collected without the results making any real difference, and may also be the cause for some participants being disinterested in voicing more concerns or trying to improve the situation.

Some participants mention already existing systems for giving feedback on specific SCA tool warnings. For example, some know of the possibility in the Coverity UI to mark warnings as false positives (and thus turn them off) and to leave comments on specific warnings. *I6* uses this frequently, though he does not seem to count it as a feedback system. *I3*, who cannot use this function anymore because of withdrawn Coverity licenses, says it would be good to still have this feature.

5.1.3 T3: Experiences and thoughts on competition

The competition culture at Axis was examined in order to see if game elements with competition aspects could be appreciated or not. This section goes through the results, as well as participants' thoughts on privacy issues in the context of gamification and competitions.

Competition culture

When asked if they enjoy competitions, around half of the interviewees are positive, while others are mildly or strongly against it. Some claim that this is due to personal preferences since they are simply not competitive people, while *I3* says he appreciates competition in general but not regarding work-related things. *"[...] in the job I think that competing against each other, it shouldn't be competition, instead we ... you are like a team that delivers a product, and then one shouldn't be like: 'Ah, you have solved the most bugs this week"*(*I3*).

Currently, there seems to be little to no competition culture at Axis. When asked about competitions at work, participants mainly mention happenings outside of work hours, such as AW events, and smaller competitions that are not directly connected to work tasks. *"We have surely had something 'silly', but no [laughs]. It is more like, what should this project be named, or so. I can't think of anything else"* (*I6*). The Axis company culture, or the culture within specific teams, might even be generally opposed to competition. *I6* says: *"We are very, very prestigeless in our team,"* and *I5* says that he prefers finding ways to collaborate than to compete.

However, this is at least partly contradicted by the competition story of Advent of Code, which was commonly brought up during the interviews, and which is described in Section 2.4.1. All interviewees had heard of Advent of Code and knew people who had participated, and some interviewees had participated themselves. The general thought seems to be that Advent of Code seems fun, but with the major caveat that it is difficult to find time for it since it has to be done outside of working hours - some say they would be more inclined to participate if it could be done during working hours. However, some (though not all) participants' appreciation of Advent of Code mainly seems to lie in the problem-solving aspect rather than the competition element. *I3*, who participated once, mostly enjoyed discussing the solutions with others and understanding different ways of to solve the problems.

Competition preferences

When it comes to which types of competition setups the participants would prefer (company-wide, teams-wide, team-vs-team, or similar), the opinions also differ. Multiple participants say they would enjoy competing within the team or as a team against other teams most; these participants enjoy competitions more when they know the ones they compete against, or together with. *I1* notes that company-level competitions would be less personal and *I4* thinks it would be less motivating to compete against a lot of people, with a much smaller chance of winning. Though, noteworthy, *I5* would prefer competitions at an organizational level. *"I think it's better as an individual on a company level and not within the team. I immediately become scared that it would create tensions and so within the team. But it depends a lot of course on what exactly it would be"* (*I5*).

Some participants seem much more positive about the idea of team-vs-team competitions, either because they enjoy collaboration, and/or because they would like to get more contact with other teams. Some seem positive about the idea of using such competitions

(and intra-team competitions) as team-building activities, especially the collaboration factor. Others do not appreciate team-vs-team competitions and seem to think it would be detrimental to team-building. *"It leans toward a 'no'. It becomes a little 'we against them' feeling. As I said before, we will still deliver the same product. One wants to be, like, 'one', a large group of teams"* (*I3*).

All interviewees care a lot about the purpose and goal of the competition, though in two very different ways: Some highlight that they want the competition to regard only useful things that encourage work efficiency and better practices - others do not want the competitions to be work-related at all. As a voice of the first sentiment, *I7* is not very drawn to competition but is positive if it can motivate others to, for example, use SCA tools more; though, he himself does not need further motivation. Also *I6*, who generally is against competitions relating to work, says that team-vs-team competitions in using SCA tools might be useful if they promoted good SCA practices by showing some teams that other teams work successfully with for example Coverity. Meanwhile, other participants do not want competitions to relate to work tasks. One reason for this is the risk to foster a more competitive culture rather than a collaborative one. *"But if one separates it from the normal product development and says 'Today we're going to have some kind of competition', then I would have thought it was fun"* (*I8*). When it comes to Advent of Code, it seems to hit a middle ground between the two views found in the interviews, that what is done during working hours should be relevant for work, and that competitions should not relate to work-related tasks. *"It's very fun. A little separated from work, but still related. So then it's fine"* (*I3*).

Concerns about competitions

Multiple participants also brought up more specific worries about how work-related competitions can be difficult to set up, and can give unfair results.

One concern is that it can be difficult to measure relevant metrics, for example for sorting a leaderboard - it could be easy for results to become misleading and unfair. This could be especially true for larger competitions, involving multiple teams, because of the differences in tool setup, work processes, and code bases. During the interviews, the example given by the interviewer was of competing to get the highest number of solved bugs. This was met with skepticism among most participants, partly because of the assumed difficulties to find reliable metrics to compare participants. For example, as pointed out by *I3*, some bugs just take longer to solve than others.

Further, there is a worry among some participants that competitions, when connected to work-related tasks, could motivate wrong behaviors if the competition is poorly designed. Developers may start to focus on optimizing work for the sake of the competition, and disregard other important aspects, which could be detrimental to overall work efficiency and results.

Another concern mentioned by multiple participants is that competitions can be stressful. *I6* notes, regarding Advent of Code: *"It must not be so that one feels pressure to do it because it's part of the job. I think it's more of a private thing that it's fun"* (*I6*).

Privacy and anonymity

Regarding anonymity, all participants say they are comfortable with sharing data such as competition points. The Axis culture builds on openness, which has shown itself throughout the interviews. *I5* says that he is used to sharing all his data at work, thus it would not feel bad to share some more. He notes that specifically, data about SCA interaction does not feel that private, a view shared with multiple other participants. However, as noted by *I6*: *"I think you will find the whole scale on Axis, privacy and not privacy."*

Especially, participants say they have no problem with sharing data within the team. When it comes to sharing on a company level, most are equally okay with this, while some have minor concerns. *I1* mentions, even though he would be a little less comfortable with this type of data sharing: *"But at the same time, then there would be so many numbers so no one probably cares what one's score is."*

However, some interviewees brought up the risk of competition data becoming a performance indicator, which they often tied to the question about privacy and anonymity. *I1* mention that others (though not him) might have concerns about privacy issues even for data shared within the team if it could become a score of how much one contributes. *I8* thinks that this could be more of a concern when sharing data with the whole organization since competitors might not know how the data was used and if it would affect one's position at Axis. *"I don't know the whole company as well [as I know my team] and I don't have the same trust that no one would think of basing salaries on some indicator"*(*I8*). *I8* is, however, still positive about the idea of sharing such data with the whole company.

When it comes to avatars, as a way to achieve anonymity, there are arguments both for and against them. *I7* notes that avatars make it more difficult to get in contact with someone if one has a question or similar. *I8* says that anonymity might be more comfortable, but also less motivating if one is motivated by the leaderboard. The participants are overall split regarding if they would want to use the anonymity function or not. Some say they would not use it, but that it is a good feature for those that feel more comfortable that way. *I5*, meanwhile, thinks avatars would feel weird within the company: *"If there is something that is so sensitive, it's probably better to not have it at all, I think. In practice it feels like it will surface anyways, it's a pretty small organization, after all."*

Multiple participants mentioned the importance of having leaderboards, or other similar data sharing, as an opt-in-opt-out feature. As mentioned by *I4*, it is not guaranteed that everyone, or himself, appreciates the competition and the publicity of the results at all times, so there should be an option to opt-out in periods. *"It's a little weird if one is new at the job, and is, like, last at the leaderboard. (Laughs.) Doesn't feel that fun, perhaps"* (*I4*).

5.1.4 T4: Motivation by work efficiency and work results

Another overarching theme in the interviews, which was visible throughout all the different interview sections, is the tendency of the participants to value work results and efficiency over for example playfulness and fun. This is however mostly tied to the actual development work, which the developers want as fricton-free as possible - the development itself, and seeing the results of it, seems to be enough reward for their work to be motivating.

Prioritizing work efficiency and results

In general, developers seem hesitant to the idea of gamification. Participants seem to think that their work, also regarding SCA tools, is already rewarding enough, without the need of making it more fun or adding rewards. *"Now the reward of the tools we use is perhaps to find bugs. That, after all, has to be the primary reward"* (*I2*). *I2* notes that if the developer that puts in the work feels like they do not get any reward for it, the used tool must be bad, and these problems should be evaluated rather than adding gamification on top of it. This indicates that he thinks development tasks should be, and often are, intrinsically motivating enough by themselves. However, he agrees that this may not be the case for less motivating tasks such as leaving feedback or crowd-sourcing information.

Also, participants often mention the importance of work efficiency and tools that are quick and easy to use - for example, they do not want to do multiple clicks to reach the information they need. Regarding giving feedback or viewing feedback on a separate page, *I5* says: *"That becomes an additional [...] separate step that doesn't really bring my work forward and doesn't do so that I can deliver results quicker."*

When it comes to prioritization of work results, this was often brought up when asking participants about their views on competitions at work. Some mention they would be positive to competition if it could motivate people to increase the quality of the code; others, on the contrary, are hesitant because they do not see a clear connection between their work and competitions, indicating that they mainly want to do work-related things during working hours, and are motivated by this. *I6* says: *"I don't do it for ending up at some leaderboard, I do it for us to deliver good things."* Still, this can be viewed against the fact that all participants (of those asked this specific question) said they would appreciate Advent of Code more if it could be done during working hours.

This theme is interesting seeing that some participants clearly are not intrinsically motivated enough to use SCA tools, to give feedback, or to try and improve the situation. Still, all participants seem mainly motivated by work quality and efficiency. Those positive to SCA tools say that their motivation for using SCA is the increased code quality - meanwhile, those not using SCA seem to note no deterioration of quality, and instead are motivated by efficiency to not use the tools.

Opinions about playful user interface

One finding that may be connected to work efficiency prioritization is that participants are generally skeptical of the idea of a playful user interface in development tools. They mostly want tools to be simple and easy to use, and think a playful user interface would be in the way. *I8* explicitly states that the dominating motivation in his work is to make the products better, and thus a playful user interface would not motivate him more. *I5* says that he probably quickly would start to disregard the playfulness aspects of the tool. *"I like GUI's that quickly give an overview and correct information, even if it's not always the prettiest"* (*I5*). This can also be linked to the fact that multiple participants say they prefer to use the terminal as their primary development tool.

Overall, the participants seem to want development and work tools to be serious and professional and do not seem to see any purpose for a playful user interface. *I4* says he wants to be able to take a tool seriously, which he cannot truly do if it has a playful user interface. As an example, he mentioned the Python formatting tool Black. *"[Black] always gives some*

emojis when something goes good or bad. It doesn't always feel that serious with a tool that gives a cake when you click on it. [...] It feels like it doesn't really fit in" (*I4*). Others have similar views. *"It's a little depending on how one does it, of course, but there is perhaps a limit where one thinks that now it's just silly. [...] One doesn't want it to look like some kind of mobile game, with a lot of flashy things"* (*I1*). Both *I1* and *I5* label themselves as *"rather boring"* and think in general they would not appreciate playful design.

When questioned if they would appreciate playfulness if it hypothetically did not hinder work efficiency, for example if it resided on a separate page not connected to the development setup, individual preferences differ. Some say they may appreciate it and probably would have found it fun, others think they would be indifferent and not use visit this separate page often. *"It is more if one should present for someone else, then it's useful. If it facilitates visualizing something that can help people understand. But if it's some funny pictures or something that should spread joy in everyday life ... I don't think that part would be super important"* (*I2*).

Still, some less direct evidence indicates a more playful culture at Axis. One example is the team names at Axis, partly discussed with one participant. The team names are often more playful than descriptive, for example, Skeletor and Hajpool, where *high* has been switched for the Swedish *haj*, meaning *shark*. As mentioned earlier, *I6* also noted that his team has had some *"silly"* competitions, for example, to decide on project names.

5.2 Questionnaire findings

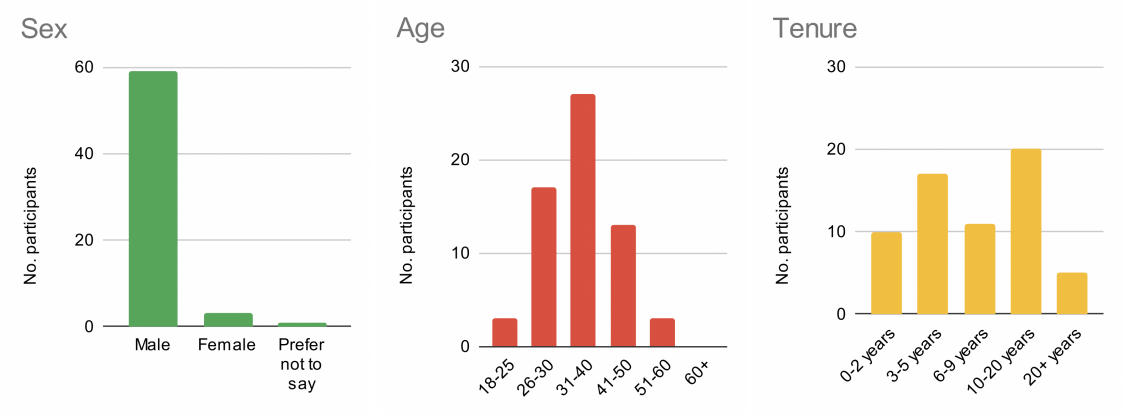
This section presents the results of the questionnaire, including employees' Hexad player types and feedback-giving preferences. A total of 63 answers were submitted.

5.2.1 Control variables

The control variables used in the questionnaire were sex, age, and tenure, and the results are shown in Figure 5.2. Notably, almost all participants were male, thus we will not be able to draw any conclusions from how results relate to sex. In terms of age, the distribution roughly follows a bell curve, where the majority of participants fall between 26 and 50 years of age, with 31–40 being the most represented age group. Regarding tenure (how long the participants have worked in development, but not necessarily at Axis), the distribution was more even. Developers with 10–20 years of experience were the most represented, while developers with 20+ years were the least represented.

In the question about participants' sex, an additional option "Other" was given, but since no participants utilized this option, it is not shown in Figure 5.2a.

The control variables were used in the analysis of the questionnaire to see if there were major differences between how groups answered the questions. No variations are displayed in either of the following sections, as no major differences were found that could not be explained by individual preferences and/or the sample sizes within different control variable categories.



(a) Sex distribution. **(b)** Age distribution. **(c)** Tenure distribution.

Figure 5.2: Distributions of control variable results from the questionnaire.

5.2.2 Hexad player types

The Hexad player types were first evaluated by calculating players' primary player type, as suggested by the creators of the framework [35], [37], and the data was then further examined to give a more nuanced view.

Primary player types

A user's primary player type is defined as the type for which they got the highest number of points. The results of this are presented in Figure 5.3. We introduced the following additional terms: Dual, for participants with two primary types (the same score on two types), and Mixed, for participants with three or more primary types.

The most common player type was Achiever, with 39.7% participants belonging to this type, and Philanthropist in second place with 27.0%. The least common primary player types were Player with 3.2% and Disruptor with 1.6%. Dual and Mixed types consisted of 19.0% percent in total.

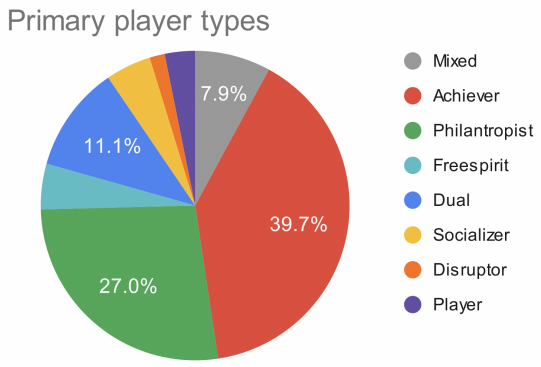
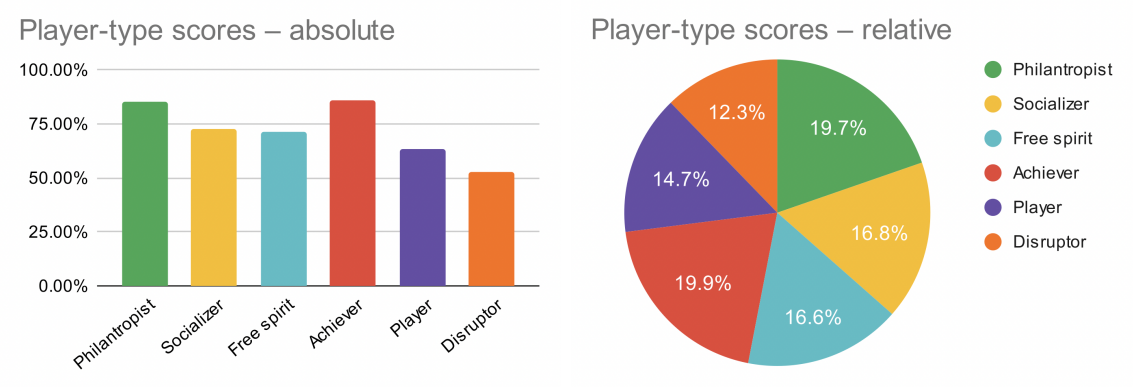


Figure 5.3: Percentage of participants' primary player types.

Group score distribution

Although the Hexad framework [37] only looks at primary player types, this may lead to a loss of certain nuances, for example since a person's primary player type may only be a single point away from their secondary type. Therefore, we also present the raw data on player scores at a group level. Figure 5.4a shows the absolute percentages of each player type, and Figure 5.4b clarifies how the absolute scores relate to each other.



(a) Absolute percentages of the maximum achievable score reached for each player type.

(b) Relative percentages between player-type scores.

Figure 5.4: Player type scores at a group level.

Similar to the primary player type results, the Achiever and Philanthropist types are the most represented. However, the raw data does not show such a clear distinction as for the primary player types; participants at a group level score rather similarly for all player types, and the Player and Disruptor types are no longer underrepresented. The differences in results may be explained, at least in part, by the Dual and Mixed types being factored into this data.

Score variation

In order to further nuance the results, we also present data on the score variation between participants within each player type. A larger score variation means that some participants may strongly adhere to this player type while others do not at all relate to it, making the previously presented group average a less reliable metric for drawing conclusions about individual preferences.

Figure 5.5 plots the respondents' score variation for each player type. It shows that the Philanthropist and Achiever types have relatively low variation among participants, while the Player and Disruptor vary significantly more.

5.2.3 Feedback on SCA warnings

The following section contains the results of the third section of the questionnaire, regarding developers' motivation to leave feedback on SCA-tool warnings, which is given in the Appendix C; all the questions were answered on a 7-point Likert scale. Note that all questions in this section were optional for respondents to answer, and the response rate varied between questions (on average 60.4 respondents, with a minimum of 59 and a maximum of 61).

Note also that the results are often presented as the distribution between answers on the Likert scale, where scores 1-3 are reported as "negative", score 4 as "neutral", and scores 5-7 as "positive". A group that is "more positive" than the average means it had a higher percentage of participants belonging to the score group 5–7 but does not necessarily mean it had more in the 1–3 group (this is analogously for "less positive", "less negative", and "more negative").

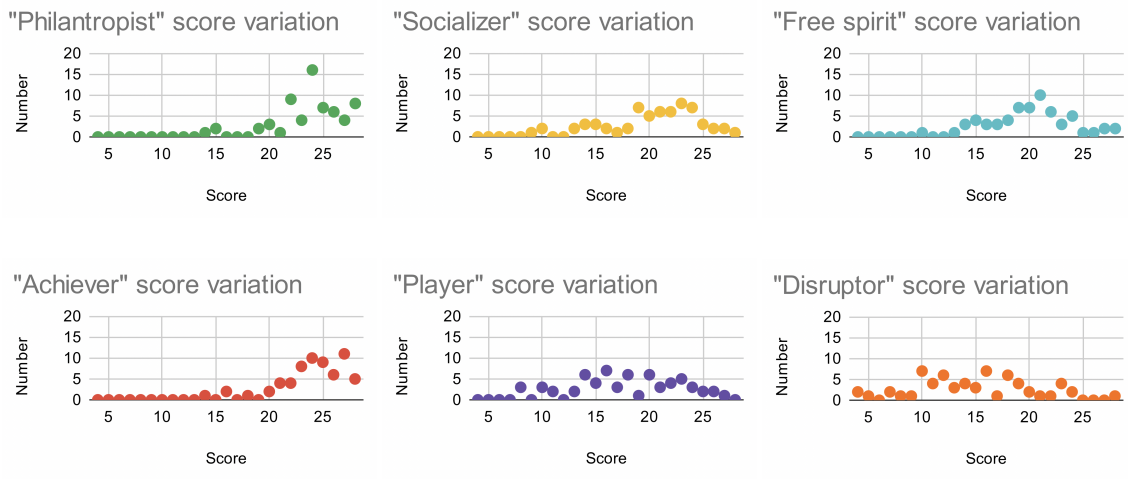


Figure 5.5: Score variation for each player type. The x-axis ranges between the minimum possible score (4) and the maximum possible score (28). The y-axis shows the number of participants that got a certain score.

Motivation to view and leave feedback

The results regarding if users would like to leave feedback, and/or be able to see others' feedback on SCA warnings, are presented in Figure 5.6. 52.5% of participants were positive about giving feedback on SCA warnings, while 26.2% were not motivated to do this; 21.3% were indifferent. When it comes to viewing feedback posted by others, developers were even more inclined, with 70.0% being different levels of positive and only 16.7% being uninterested.

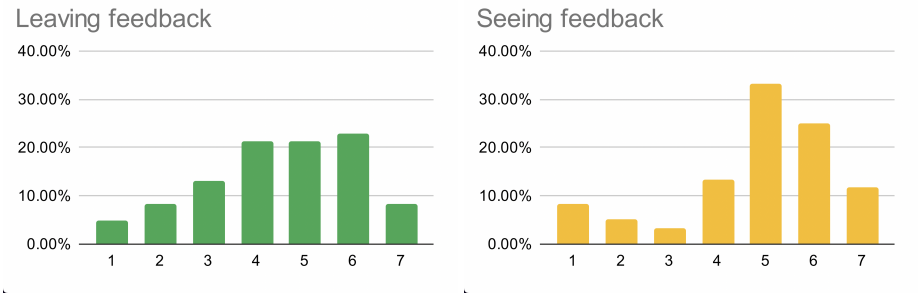


Figure 5.6: Motivations for leaving/seeing feedback on SCA tool warnings, on a 7-point Likert scale.

Motivation for leaving different types of feedback

Figure 5.7 presents the results for developers' motivations to give multiple-choice feedback (e.g., marking a warning as a false positive), and text feedback (e.g., describing why a warning is a false positive). Overall, developers seem more motivated to give multiple-choice feedback (62.3% were positive to this), while the corresponding value for text feedback is lower, at

39.3%. Participants were also more often directly negative to text feedback than multiple-choice feedback. In both cases, the indifference rate was 14.7%.

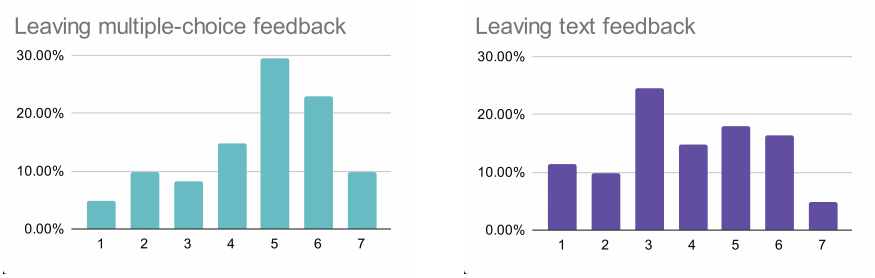


Figure 5.7: Motivations for leaving different kinds of feedback on SCA-tool warnings, on a 7-point Likert scale.

Motivation by helping and improving

Respondents were asked about two possible motivators for giving feedback: the possibility to help other users (without specification of how this would be done), and the possibility of directly improving the system. The results are given in Figure 5.8. The vast majority of participants reported that they would be more motivated to leave feedback both for helping other users and for improving the system, with positive scores of 85.2% and 88.1%, respectively. Notably, more respondents felt very strongly about improving the system, in comparison with how many felt very strongly about helping other users. Approximately 10% claimed they would not become more motivated by these factors (9.8% regarding helping users, and 10.2% regarding improving the system). 4.9% of participants were indifferent when it came to helping other users, while only 1.7% felt indifferent regarding directly improving the system via feedback.

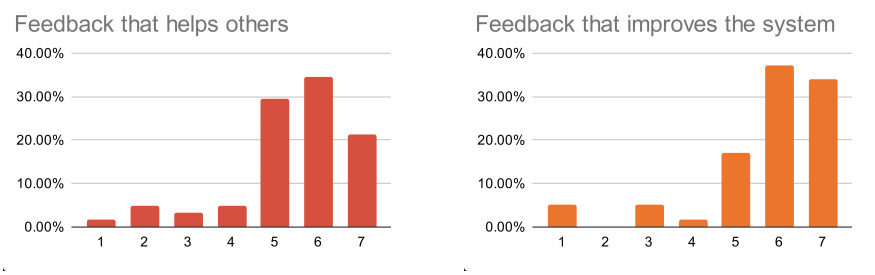


Figure 5.8: Whether developers would be more motivated to leave feedback depending on different factors, answered on a 7-point Likert scale.

Motivation depending on previous feedback

Respondents were also asked if they would feel more motivated to give feedback if there was no previous feedback given, or if there was previous feedback either with a similar opinion as their own or a different opinion from their own. As before, the example provided to

the respondents was that they were noting whether a warning was a false positive or a true positive.

The results are given in Figure 5.9. When it comes to leaving feedback with no previous feedback given on the topic, answers were only marginally positive and generally quite distributed around indifferent values (49.2% positive, 27.9% negative, and 22.9% indifferent). A quite similar result was given regarding the case where it existed previous feedback of the same opinion as that of the developer, with varied results slightly similar to a normal distribution around the middle (38.3% positive, 35% negative, and 26.7% indifferent). Comparatively, developers seem much more motivated to give feedback if there exists previous feedback of differing opinions than there own (68.3% positive and 18.3% negative).

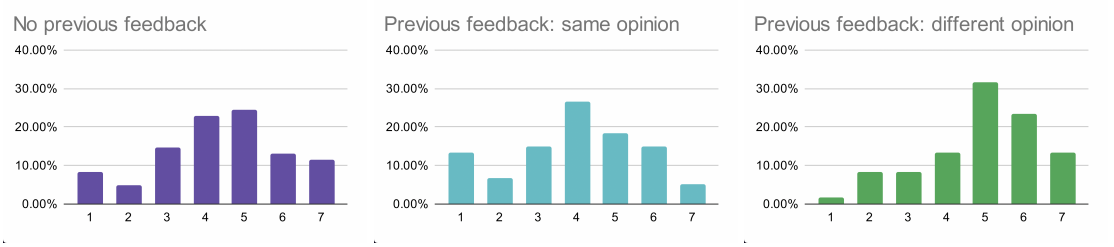


Figure 5.9: Whether developers would feel more motivated to give feedback, depending on the previous status of provided feedback on the topic. Answered on a 7-point Likert scale.

5.3 Conclusions of user research

The user research was conducted based on the goals listed in Section 3.2.1, which have all been fulfilled and for which the results have been described above. This section lists some overall conclusions from the user research which will guide our system design. All conclusions are about developers at Axis and are not necessarily applicable to developers within a different company culture.

- Opinions on SCA tools vary greatly between individuals and teams, mainly based on repository-specific SCA problems and opinions on tool usefulness.
- Developers are irritated by multiple SCA usability issues, e.g., configuration problems, false positives, and slow UI.
- Developers often ignore warnings they find not useful. Especially developers with a dislike for some SCA tools often find ways to ignore or disable the whole tool.
- Most developers find SCA warnings generally easy to understand and fix, with the exception of suspected false positives, which takes longer time to research.
- Developers already positive to SCA sometimes take the initiative to use more SCA tools, while more negatively inclined developers generally do not take the initiative to find better SCA tools, and instead prefer using no SCA tools.

- Most developers would like the option to turn off warnings in tools (which is often already possible), though some acknowledge a risk with turning off useful warnings.
- The majority of developers would like to be able to give feedback on SCA-tool warnings, and even more would like to be able to see others' feedback. Seeing others' feedback seems more appealing than giving feedback oneself.
- Developers are more motivated to leave multiple-choice feedback than text feedback.
- Developers are much more motivated to leave feedback if they feel like they are improving the system or helping other users.
- Developers are more motivated to leave feedback if feedback already exists but is of a different opinion than theirs.
- Developers highly value work efficiency and the quality of their work. They want their setup to be efficient and serious, without extra steps needed to carry out their work.
- Developers' opinions on competitions vary greatly; because of a common suspicion against competitions, they must be well-designed, not used as a performance indicator, and preferably be opt-in and opt-out.
- Developers enjoy challenges and problem-solving.
- Developers in general are willing to share gamification data and do not find this to be sensitive data, though they acknowledge the value of anonymity for those that want to have it.
- All different Hexad player types are well-represented among the developers, however, people are more united about supporting the Achiever and Philanthropist roles, and less united when it comes to Disruptors and Players. Socializers and Free spirits rank somewhere in between this.

Chapter 6
System design

This chapter describes the final system design and the decisions (based on the previous literature review and user research) that led to this design.

The system's main purpose is to provide a way for developers to give feedback on Coverity, both on the general analyses and on specific warnings. By aiming for a suitable feedback collection design, the goal was to make the system optimal for collecting as much user feedback as possible. Additionally, we designed gamification features that could be added to the system, in order to examine if these would increase or in other ways affect the submitted feedback.

6.1 Technical system design

This section presents an overview of the technical design of the feedback system, which consists of many different components and communicating parts; Figure 6.1 gives an overview. The main four system components are: a function added to a pre-submit Jenkins job, a Gerrit plugin, an Elastic Stack database with Kibana visualizations, and a Confluence space for the users to view the feedback results and game data.

6.1.1 Jenkins code

At Axis, all commits pushed to Gerrit will automatically trigger a pre-submit job that is run in Jenkins. This job may, depending on the type of code in the repository and the commit, include running a Coverity analysis. In this already existing pre-submit job, written in Groovy, we added a function that is triggered if a Coverity analysis has been run. The function sends an SSH call to our Gerrit plugin, enclosing the result of the Coverity analysis and metadata about the patchset.

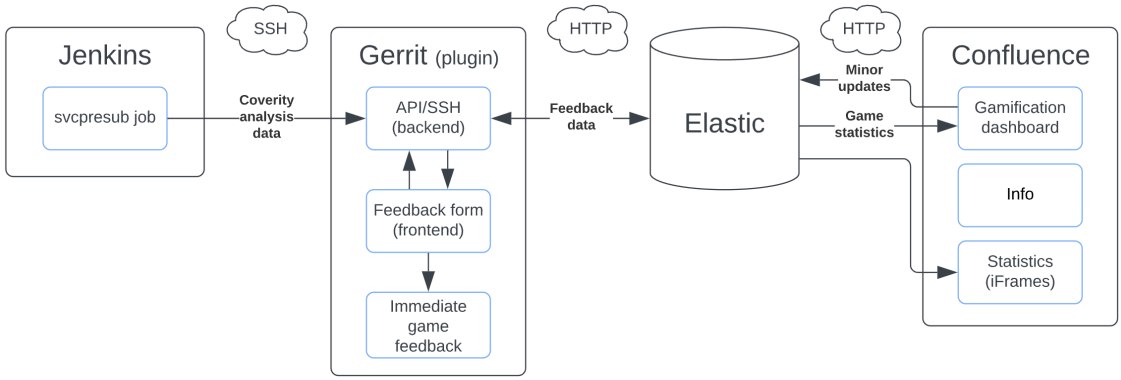


Figure 6.1: An overview of the components and data flow in the system design. The clouds show the communication protocol between parts.

6.1.2 Gerrit plugin

The Gerrit plugin is the central part of the feedback system and was deployed on the Axis Gerrit site. It was written in Java and Javascript, using the LIT framework. It consists of a frontend custom element that is added to the Gerrit UI (more details given in Section 6.2.1), and a backend that handles gamification logic and connections to the Elastic database (via the Java Elasticsearch API, which sends HTTP requests to the database). The plugin also includes a REST API for frontend support and a command module that is accessible through SSH.

The plugin's main functionality is to make it possible for developers to give feedback on Coverity directly via Gerrit. The plugin adds a feedback section to all changes in Gerrit for which a Coverity analysis has been run. The section holds some information about the experiment (including a link to the Confluence page) and a feedback form. The form allows the user to give general feedback on the Coverity analysis done on this patchset and feedback on each Coverity issue if the analysis generated any issues. The user can also note if they found any issues in the code that Coverity overlooked. Feedback can separately be given for each patchset within the change since a Coverity analysis may be run for multiple patchsets. On form submission, the plugin posts the feedback data (and calculates and posts gamification data if the user is a gamification user) to the Elastic database.

The Jenkins pre-submit job calls the Gerrit plugin's command module; when an SSH call is received, the plugin posts the received patchset and Coverity analysis data to the Elastic database. The reason for the plugin handling this task (rather than pushing the data directly from the pre-submit job) is that this was implementation-wise easier than integrating an Elastic connection into the Jenkins environment.

6.1.3 Elastic database

The database resides in an Elastic deployment exclusively used for this system. The deployment consists of multiple indices, storing general data, gamification user data, and metadata for handling gamification tasks. Each index is shortly described below.

Patchsets stores data about the patchsets in Gerrit for which a Coverity analysis has been run (for patchsets pushed during the feedback system deployment time). The data

consists of patchset metadata and Coverity analysis results, including any issues the analysis might have flagged.

Feedback stores the feedback submitted by users.

Total-user-data stores the overall data for each gamification user: their points, achieved challenges, and count of submitted feedback forms. This is for the convenience of quickly fetching user data.

Points is a more fine-grained store of gamification events: One entry is created each time some points are awarded to a gamification user, also including data on why the points were received. This is to allow more fine-grained data evaluation.

Challenges stores static metadata about the gamification challenges.

Milestones stores static metadata about the gamification milestones.

Cookies stores data about the cookie codes used in the gamification setting. For example, it is noted for each cookie if it has been distributed to a user or not.

6.1.4 Confluence space

For providing info about the experiment and the results, as well as functioning as a game dashboard, a Confluence space was created at the Axis Confluence server. The space was made visible to all Axis employees, including all participants in the experiment, though some pages were only visible to users in our gamification test group. In addition to an overview front page, the space consists of three main parts: info pages, statistic pages, and a game dashboard.

The info pages give the user information about this thesis, how the feedback data is collected and handled, and the gamification game rules (which are supposed to be self-explanatory but might not always be).

The statistic pages present data about the Coverity analysis runs, how much feedback has been collected, and feedback results. This is presented in Kibana dashboards (created on the Elastic deployment) that are included via iFrames. Additionally, the statistics pages present a table that lists all Coverity issues discovered since the start of the feedback tool deployment.

The game dashboard is only visible to gamification users and is user-specific. It shows the current viewers' gamification data, including, for example, the user's points, leaderboard position, and achieved challenges. This functionality is achieved by including Javascript scripts directly on the Confluence page; the scripts fetch game data from the database and also do minor database updates connected to the user's cookie code management.

Confluence was chosen for these features, instead of hosting a separate web page, since its use at Axis is already widespread. We wanted to reach developers in environments they were already used to, in order for the system to seem more natural in the development workflow.

6.2 Feedback interface design

This section presents the interface designs for the feedback form in Gerrit and the feedback displays on the Confluence space. We review the different UI components, present example

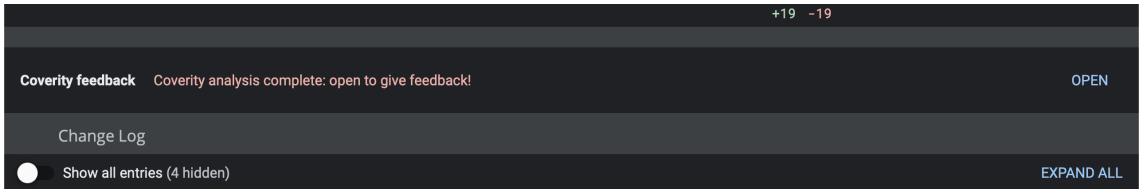


Figure 6.2: Collapsed view of the Coverity feedback section in Gerrit.

figures, and refer back to the literature study and user research to motivate design choices.

6.2.1 Feedback form

The feedback form is the main frontend part of the Gerrit plugin. This section goes over all the UI components of this interface.

Overview

The feedback form is added as a small section on relevant Gerrit change pages, in between the *Files* and the *Change Log* section on the page. Similarly to items in the change log, the feedback form is expandable when users click on its top part (buttons to open and close the forms were also added for clarification). The feedback section is available on all changes where at least one patchset has an existing Coverity analysis in our database.

An example of the collapsed feedback section is shown in Figure 6.2. When collapsed, the section only displays a short status message, describing the Coverity analysis state of the latest uploaded patchset. There are a number of different possible statuses: If no analysis has (yet) been made on this patchset (but on earlier patchsets), if an analysis was run but encountered an analysis error, or if there has been a Coverity analysis on the patchset and feedback can be given.

Once the feedback section is expanded, the user is presented with some information, as well as the actual feedback form. The feedback form consists of a general feedback section and two sections for issue-specific feedback. Figure 6.3 gives an overview picture of the expanded form.

If the Coverity analysis ran correctly, and the user has not previously submitted feedback on the selected patchset, the feedback form will be visible immediately (as in Figure 6.3). At the end of the form, there is a *Submit* button (see (7) in Figure 6.3); to submit a form, at least one form value must be given – this could be any input, as long as the form is not empty. Upon submission, the form will be hidden and exchanged for a button that allows the user to submit additional feedback – pressing this button will re-open the feedback form.

Information

The first part of the expanded feedback section (see (1) in Figure 6.3) contains some information for the user. It is noted that not all fields must be filled out to submit the form and that any feedback provided will be visible (in an anonymous manner) on our Confluence page. The link to the Confluence page is also included.

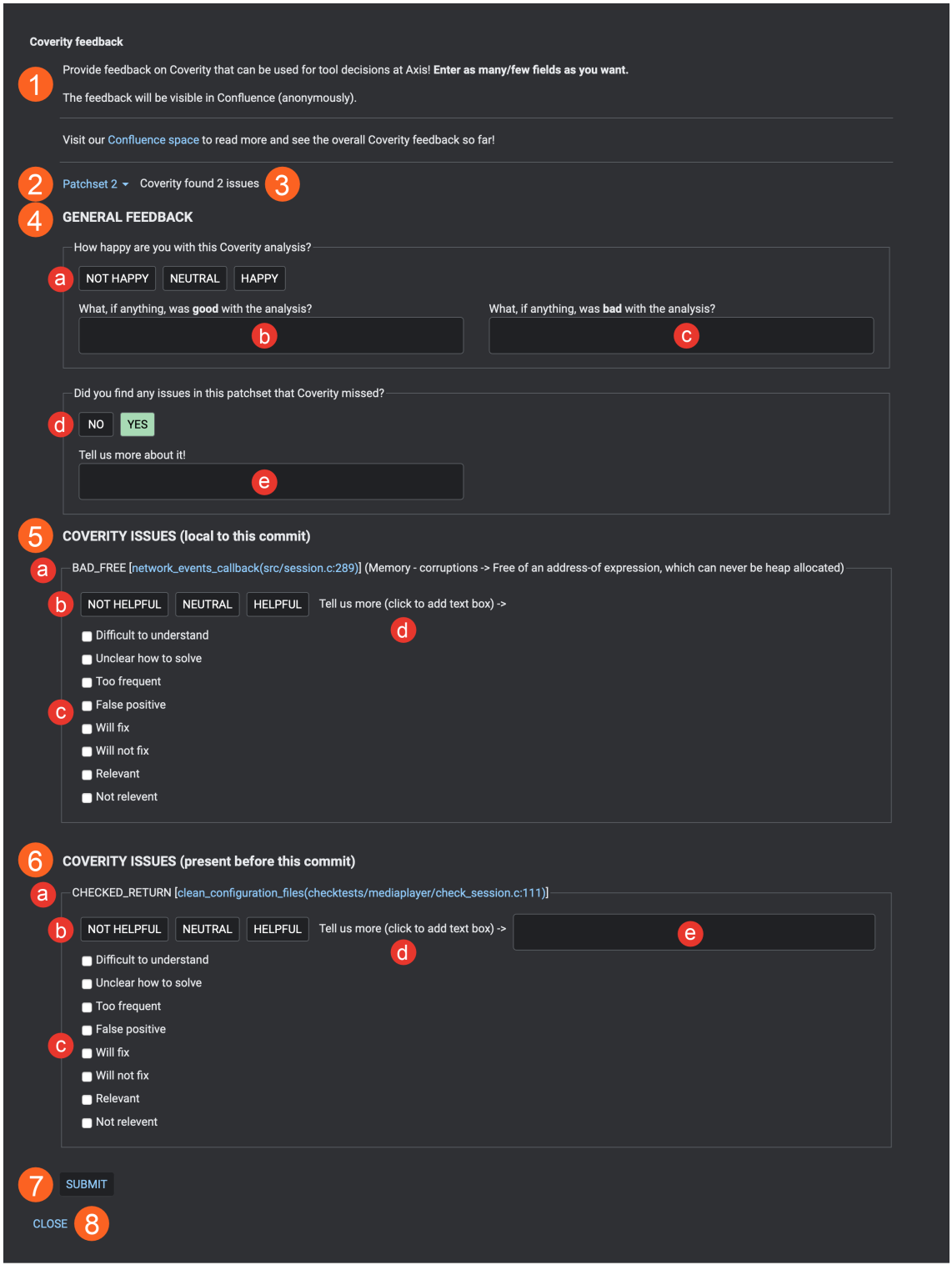


Figure 6.3: Detailed view of the expanded feedback form with all UI elements marked.

If the user is a gamification user and has saturated the number of points that can be acquired by giving feedback on this specific change, an additional message is given here: *(You can still submit feedback on this change, but cannot get more points).*

Patchset drop-down menu

Since Coverity analysis results, and the feedback on these results, are associated with a specific patchset and not a whole change, we also added a drop-down menu where the chosen patchset can be selected (see (2) in Figure 6.3). The drop-down is not connected to the similar drop-down in the *Files* section of the Gerrit change page, since we wanted to keep all plugin UI components close to each other, for clarity and consistency.

Next to the patchset drop-down menu, a status message is displayed (see (3) in Figure 6.3), which is similar to the status message on the collapsed form but dependent on the analysis state of the selected patchset instead of the newest patchset. The status shows the result of the Coverity analysis and how many issues Coverity found (0 if no issues were found).

General feedback

The general feedback section (see (4) in Figure 6.3) is visible if the Coverity analysis passed (no issues found) or failed (issues found). The purpose of the section is to gather data about the general opinion of Coverity.

The first question (*How happy are you with this Coverity analysis?*) has two inputs. First, three radio buttons ((4a) in Figure 6.3) for providing quick feedback about the overall satisfaction of the analysis (*Not happy*, *Neutral*, *Happy*). Then, two free-text fields (4b, 4c) for allowing more detailed positive and negative feedback about the analysis (*What, if anything, was good/bad with the analysis?*). The second question (*Did you find any issues in this patchset that Coverity missed?*) also has two inputs, consisting of two radio buttons (4d), and a hidden free-text field (4e) that is expanded if the user clicks the "Yes" alternative on the radio buttons.

Issue-specific feedback

If Coverity found issues for the patchset viewed in Gerrit, the feedback form shows a section for each issue, where issue-specific feedback can be given. The issues are organized into two categories: issues local to this change and issues that were present in the git already before the patchset (see (5) and (6) in Figure 6.3). This categorization was made to give the user a better overview, and because the issue types differ regarding the amount of information available from the Coverity analysis. All issues are described by the file, function, and line they occurred in, as well as the checker name (e.g., BAD_FREE) (see (5a) and (6a) in Figure 6.3), and more information is displayed for local issues (see (5a)). The issue hyperlinks go to an HTML page containing the Coverity analysis stack trace for the relevant issue. This page is already presented in other places connected to Coverity analysis results and is thus already well-known to users.

For each issue, the user can give feedback via multiple inputs. First, there are three radio buttons ((5b) and (6b) in Figure6.3), for providing quick feedback on whether the issue was helpful (*Not helpful*, *Neutral*, *Helpful*). Then, there are multiple-choice buttons (5c, 6c) for

tagging the issue with different categories. There is also a hidden free-text field (6e) where more detailed feedback can be added. This field is displayed if the user clicks on a text prompt (5d, 6d).

The options for the multiple-choice buttons were selected by combining common critiques on issues found in SCA tool literature and our user research. The complete list of options and the tooltips visible when hovering the checkbox text are presented below. See also (5c) and (6c) in Figure 6.3.

Difficult to understand. This issue was difficult to understand.

Unclear how to solve. This issue did not make it clear how to solve it.

Too frequent. I see this issue too frequently.

False positive. This issue seems like a false positive.

Will fix. I will fix/have fixed this issue.

Will not fix. I will not fix this issue.

Relevant. This issue is relevant for code improvement.

Not relevant. This issue is not relevant due to code specifics.

6.2.2 Design motivations

This section expands on and motivates some design choices that were made regarding the feedback interface.

Functional decisions

The perhaps most important decision about the feedback form was the integration point, as we wanted to be as close to the user's working environment as possible, without being intrusive in the development workflow or setup. The feedback research presented in Section 4.2 also suggests that it is important to keep the feedback functionality close to the users and provide the possibility to leave feedback online while using the application in question.

Both multiple-choice and full-text feedback options were included in the form. According to our user research, developers often feel more motivated to provide multiple-choice feedback, and this also gives easier data to process. Meanwhile, full-text feedback allows more exact insights. Literature on feedback collection (see Section 4.2) also noted that using both qualitative and quantitative feedback may be the most popular among users. One study also suggested hiding things like text fields if the user is likely to not have an opinion about the question, which we also took into consideration.

Usability decisions

As suggested by research (see Section 4.2), users may be disturbed if the request for feedback interrupts their workflow, or they are prompted too often; our feedback form is always available and accessible but collapsed in order to be as non-intrusive as possible. Users do not get prompted to give feedback but are rather reminded of the possibility when they see the collapsed feedback section in Gerrit. This mitigates two of the major usability problems with feedback requests. Notably, a trade-off we made is that users may overlook the feedback form if they only look quickly at the Gerrit change page, as the collapsed form is designed to mimic the Gerrit UI design.

Some of the feedback elements are hidden and expandable, namely the issue-specific full-text field (see (6e) in Figure 6.3) and the full-text field for reporting missed issues (4e). This can be argued to save visual space, in order to not overwhelm the user - in the event of many displayed issues, having a full-text field for each one may make the form look cluttered and tedious to fill in. Based on our user research, users were also not considered likely to often find missed issues and want to elaborate on these in text. This is why these specific fields, and no others, were hidden. However, this is also a trade-off with the risk that interested users may not find the hidden fields.

Research also suggests that users may become less motivated to give feedback if the perceived effort of giving feedback is high. This was one of our reasons for hiding some fields, to make the form look more doable. The same insight also motivates the decision to not require the user to fill in all fields, so that users can submit a form with as many or as few filled-in fields as they want. The aim of this was to lower the users' threshold to give feedback. A possible risk is that users may not understand this (though stated in the information at the top of the feedback form), and then still become unmotivated by the length of the form.

Aesthetic decisions

The feedback form is designed to fit into the general Gerrit design. Gerrit usually incorporates colors into their design: Green for good, gray for neutral, and red for bad, e.g., when voting for the Code-Review task, where -2 and -1 have red buttons, 0 has gray, and +1 and +2 have green. This color scheme was followed in our feedback forms' radio buttons, as shown in Figure 6.4.

Red may also signal to the user that something requires their attention. This is used in the info text on the collapsed feedback form: The text is red if the Coverity analysis has been completed and feedback can be given, gray if no feedback can be given for the latest patchset, and green if the user has already provided feedback on the latest patchset.

6.2.3 Statistics in Confluence

The collected feedback, together with other data, could be viewed by users on the Confluence space's statistics pages, namely on the pages *Feedback results* and *Feedback statistics*. These pages each showcased one Kibana dashboard which gave an overview of the collected feedback (feedback results) and metadata about the feedback collection (feedback statistics). All statistics and data were given in an anonymous way but could be filtered on repository and other values. A small part of the feedback results dashboard is shown as an example in Figure

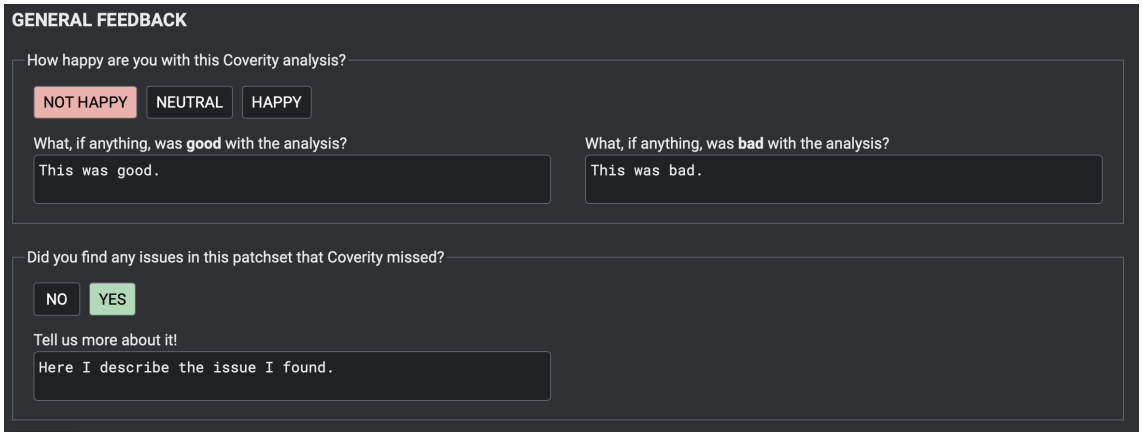


Figure 6.4: The general feedback section of the form, with example input values.

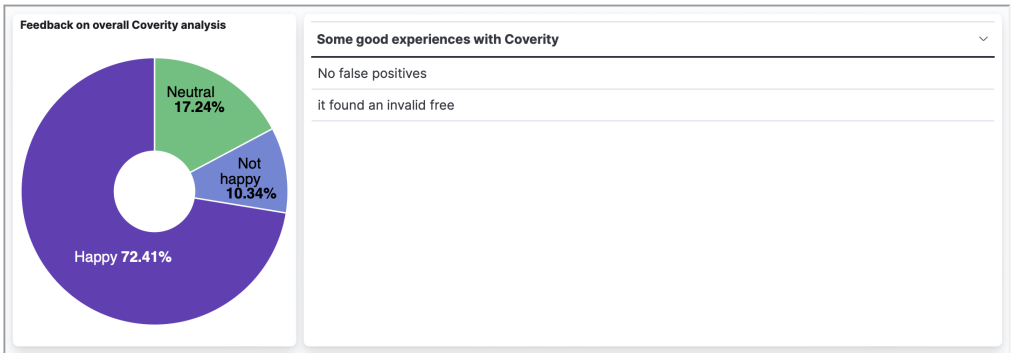


Figure 6.5: A small part of the dashboard on the Confluence feedback results page. The data shown is not relevant to the final results.

6.5.

The reason for displaying the feedback in this manner was that our user research suggests that developers are more likely to give feedback if feedback already exists, especially if it is of a different opinion than theirs. The showcased feedback may be of the same or differing opinion to an individual developer, but the idea is still that it may encourage the developer to give feedback. Here, users can also see if there is not yet any existing feedback on a specific topic, for example, their team's repository - this, according to user research, might also increase some users' motivation to give feedback (though it may decrease others).

Another finding of the user research is that developers are much more motivated to leave feedback if they feel like they are improving the system, or helping other users. While we could not directly manipulate the system based on feedback (such as by turning off warnings that are marked as false positives), we aimed to showcase that the feedback was listened to and may be of interest to others, by making it visible at least on the Confluence space.

Showing the feedback results in this direct manner might be uncomfortable for some potential feedback givers, even though the data is not mapped to their identity. We chose to still do it in this way based on previously mentioned reasons, and because the user research shows that developers at Axis generally are not overly concerned with data privacy and anonymity. However, importantly, this assumption is based on discussions in interviews around sharing

game data, such as points, and not feedback results, which may or may not be perceived as more sensitive data.

6.3 Game design

The game design consists of a few game elements, which are handled by the Gerrit plugin and Elastic database, and displayed to users on the Confluence space game dashboard (and partly in the Gerrit UI). Information about the game and game rules are provided to users via the Confluence plugin information pages, but the aim was for the game to be self-explanatory, so that information would only be sought out by curious users.

This section describes the used game elements and game rules and motivates the design decisions based on our previous literature review and user research.

6.3.1 Game elements

This section lists and describes the included game elements (both high-level and low-level), and motivates choosing these elements. To provide a clearer view of the design, examples from the Confluence game dashboard and the Gerrit UI are also given where applicable.

One consideration for the choice of game elements was the distribution of Hexad player types among the intended users (see Section 5.2.2). Note however that gamification literature (see Section 4.3) recommends including game elements that cater to players of different types, also minority types. Additionally, our user research shows that Axis developers overall (in absolute values) have similar scores for all Hexad player types. Because of this, we chose to include a set of game elements that may appeal to all different Hexad types. Still, as found in our user research, the highest-ranking player types were the Achiever and Philantropist types; they also had the overall highest score percentages and the least variation between scores. Thus, we especially wanted to include elements that catered to these types. Regarding the Player and Disruptor types, even though they had quite high overall scores, they still ranked lowest both overall and as primary player types, and had significant score variation among participants. Hence, developers might strongly agree or strongly disagree with a design made for Players or Disruptors, which made us careful with including elements only suitable for these types.

Some of the used game elements were more prevalent in the design than others, partly mirrored in the order they are presented in the list below.

Points are the backbone of the game design. A user gets points by submitting feedback in the Gerrit form. Points were used since this is a simple metric for users to keep track of, especially if they have only little interest in gamification and do not want to engage in any other game elements - points are also easy to structure other game elements around and were thus difficult to avoid including in the game design. Note however that points are mainly recommended for the Player Hexad type.

Milestones are reached when the user gets a certain number of points. In total, there were three milestones in our design, and after each reached milestone the user got an extrinsic reward. This may be argued to be similar to the game element Levels, which are recommended for both the Achiever and Player player types.

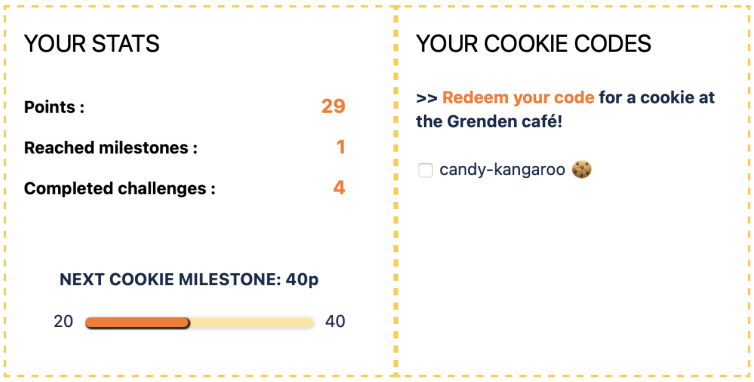


Figure 6.6: An example of the overview game metrics displayed to users on the Confluence game dashboard, including the progress bar and the cookie code display.

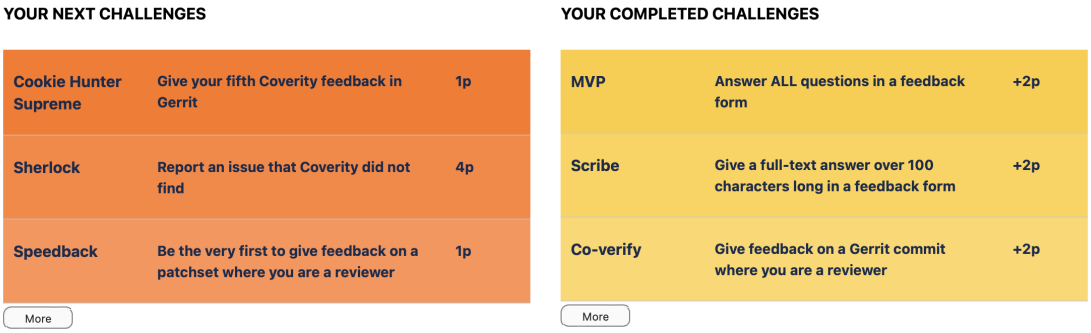


Figure 6.7: An example view of upcoming and achieved challenges for a user, at the Confluence game dashboard.

Visible progress was incorporated by displaying a progress bar on the Confluence game dashboard, showing the user their progress toward reaching the next milestone. Progress is also shown in other ways at the game dashboard, for example by displaying the user's number of achieved challenges. How the progress bar looks at the game dashboard is illustrated in Figure 6.6. Though not mentioned in the Hexad framework, we theorize that visible progress may be appreciated at least by the Achiever player type.

Challenges were included as a list of nine "tasks" for the user to achieve, all connected to giving feedback in Gerrit. When the user achieved a challenge, extra points were awarded (the number of points was challenge-dependent). The challenges were presented at the Confluence space, in two separate lists of upcoming and achieved challenges, see Figure 6.7 for an example of this view. In order to not overwhelm the user and to present a clearer overview of the UI, only a few (three) challenges were visible in the *Your next challenges* list, unless the *More* button was pressed.

Notably, according to the definition in Section 4.3.3, these tasks are more similar to quests than challenges (since they are performed without competition or time pressure), and we named them Challenges only for design purposes.

Challenges (and quests) are a suggested game element for the Achiever, Player, Free spirit, and Disruptor player types (though Disruptors might specifically like more competitive challenges).

Challenges were also included based on the finding in our user research that developers at Axis generally seem to enjoy challenges and problem-solving, for example showcased by the interest taken in Advent of Code. Also in literature (see Section 4.3), Do et al. [60] recommend focusing gamification solutions (especially regarding SCA tools) on problem-solving. However, it was difficult to design problem-solving challenges for feedback collecting, and we settled for a middle ground of challenges that were quite simple in nature. They were, however, also a way to communicate the next possible steps for users and to set up clear goals in the game.

Immediate and continuous feedback was utilized in the Gerrit plugin for showing the user when it made progress. Immediately on form submission, the user was presented with a small popup in Gerrit, describing the number of points won, as well as if any challenges or milestones had been achieved. This was done in Gerrit in order to give feedback very visibly and closely to the time of the actual achievement, instead of forcing the user to go to the Confluence space to see the points update. An example of the popup appearance is given in Figure 6.8.

Extrinsic rewards were given to a user after each reached milestone. This was in the form of *cookie codes*, which were redeemable at a café in one of the main Axis buildings, for a cookie or protein bar or similar within a price range. We choose this type of reward since it would likely be in accordance with many participants' tastes, as they could choose themselves from the café, and because it was possible for us to get a budget for it. The idea was also that this type of relaxed reward might lead away from any idea that the gamification was connected to work rewards or used as some sort of performance indicator, as worries about this were expressed during our user research.

The cookie codes consisted of randomized short phrases from a list of words on animals, colors, baked goods, and positive words. Examples are "candy-kangaroo" and "carrot-cake-nicer". See Figure 6.6 for an example of how cookie codes were displayed to users on the Confluence game dashboard.

Competition (Leaderboard) was incorporated at the Confluence space, based on the number of points the users had achieved, i.e., how much feedback they had provided. The leaderboard was anonymous and of the no-disincentive type, only showing the user's overall position on the board and the number of points needed to climb up one position (see Figure 6.9). The final five highest-ranking users on the leaderboard were promised a reward given after the completion of this project.

Social competition (which may be achieved if players, for example, compare leaderboard positions with teammates) is recommended for the Socializer, Player, and Disruptor Hexad player types; Disruptors are also likely to appreciate leaderboards. However, it was found during our user research that developers' opinions on competition vary greatly, often because of suspicion against performance indicators and against fostering a less collaborative culture. Preferably, a leaderboard should be opt-in and opt-out; instead, we choose to make it anonymous in order to not distress some users who may not appreciate the competition functionality.

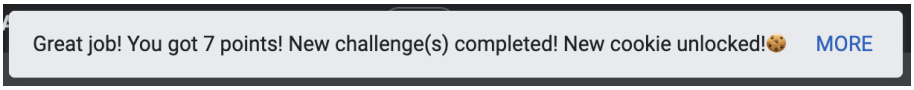


Figure 6.8: An example of the confirmation alert in Gerrit received after submitting a feedback form.

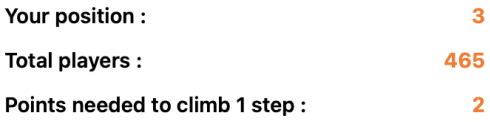


Figure 6.9: An example of the simple leaderboard on the Confluence game dashboard.

Progressive difficulty was lightly incorporated by the milestones becoming more and more difficult to reach (by increasing the needed points between milestones), and by the first displayed challenges being very easy to achieve while later ones may be harder. This was done in order to keep players interested in the game for a longer period of time.

Gifting could be said to be included as a game element since the cookie code rewards were not personal and therefore could be gifted to anyone else, e.g., other users. This possibility was also noted in the Confluence information pages for clarity. Gifting is a game element especially suited for Philanthropists.

Social graphs were included in a way by showing the general feedback statistics, and Coverity analysis results in the Confluence statistics pages (described more in Section 6.2.3). This showed feedback progress overall and provided possible filtering on a repository and similar. This could be said to adhere to "Social comparison and discovery", which is recommended for the Socializer and Player player type. However, since these statistics were visible to all users, including control group users, they can be argued to not count as a game element.

Overall, it can be noted that no visually playful user interface was included, with the exception of the cookie emojis used in connection to the cookie codes. This was because of the user research finding that developers at Axis in general want their setup to be efficient and serious. However, because of other indications of some playfulness in the company culture (such as playful team names), playfulness was instead achieved in the design at least partly by using more playful challenge names and cookie code names. The yellow and orange color palette was mainly chosen in order to have an Axis-themed layout.

6.3.2 Game rules

The rules of the gamification system were defined in an attempt to make the game as fair and enjoyable as possible. The rules were aimed to be self-explanatory (or in some cases, not necessary for the user to know about) but were also presented to curious users on the

Confluence information pages. Multiple rules are explained above in connection with the respective game elements, while the other overall game rules are listed below.

- For each submitted feedback form, a number of points were rewarded based on the number and type of the answered questions in the form. Two points were given for a full-text answer, and one point for answering a radio button or checkbox question.

- There was a max limit on the number of points a user could get from giving feedback on the same change (i.e., a collection of patchsets). This was done for fairness since some developers might push 100 patchsets on a change, resulting in many more feedback opinions than the developer that pushes only two patchsets with more careful changes. When the point limit was reached, this was displayed for the user in the feedback form as a disclaimer. However, users could always get challenge points by achieving a challenge, even if the challenge was achieved by giving feedback on a change where the point limit had been reached.

- There were only three milestones, and thus only three cookies could be won per user; this was decided in order to not exceed our budget. However, after achieving the last milestone, users could still get more points and climb on the leaderboard.

6.3.3 Moral and ethical principles

As recommended by gamification literature (see Section 4.3), we also formulated some ethical values to adhere to in our gamification design. These are listed and evaluated below.

- The ultimate, long-term goal of the system should be to improve work satisfaction among the users, not solely to reach business purposes. This could, for example, be a risk if the feedback was only collected for use in future price negotiations when buying SCA tool licenses, which might lead the users to feel manipulated and "used". However, the Axis goal of wanting feedback is to improve the SCA tool use at Axis for higher developer satisfaction since the tools are not likely to be used if developers are not satisfied. We aimed to make this goal clear to the users via the Confluence information pages.

- The gamification setting should not take unfair advantage of the users by making them do actions they do not want to or that are outside their work description. For example, we did not enable developers to rate each other's feedback (which may be uncomfortable to some) and did not include challenges that would encourage developers to do things such as work during the night.

- The gamification setting should not decrease the users' autonomy. All actions should still be, and be perceived to be, the users' choice and not forced upon them by the need for rewards or similar.

- No sensitive data (as defined by GDPR) about the users should be collected or used in the game design.

- The data usage should be mandated by a data usage policy, both for transparency and to hinder possible later improper use of the data. Information on how the data was used, and allowed to be used, was included in the information pages in the Confluence space.
- The data should not be used or evaluated for any purpose outside of the original gamification setting; for example, there should be no work-related reward/penalization out-of-game.
- All gamification aspects should be opt-out, and this should clearly be stated so that users know the possibility, and how-to, of opting out. Competition elements should also be opt-in. This was not implemented, but the overall design of the gamification was made so that a user can choose not to participate simply by taking no action and not visiting the Confluence space. Notably, gamification users cannot opt out of seeing the Gerrit popup when submitting feedback, but we did not regard this as a major issue.

Chapter 7
System evaluation results

In this chapter, we present the results of our feedback system after it was deployed for 14 days, with and without gamification. The aim of the experiment was to answer RQ5 (*When evaluated at Axis Communications, does gamification increase interest and engagement for giving feedback on static analysis tools?*), by use of the metrics introduced in Section 3.4.4: user engagement, motivation, and satisfaction.

This chapter starts with an overview of some general statistics from the experiment. Then, user engagement with the system is examined, and results are compared for the gamification group and control group; this is followed by similar evaluations of user motivation and satisfaction, though these metrics are based mostly on the follow-up chat interview findings. At the end of the chapter, there is a brief overview of the results from the feedback acquisition, i.e., the content of the feedback users gave on Coverity.

Similar to the interview participants, all chat interview respondents used he/him pronouns; for simplicity, these will thus be used when referring to participants throughout this section. Participants will also be referred to by their respective participant IDs, as listed in Table 3.2.

7.1 General statistics

At the end of the system deployment, a total number of 3317 patchsets with Coverity analysis results had been stored in the Elastic database (note that the gathering of patchsets started three days prior to the deployment of the feedback system). Of the stored patchsets, 1657 patchsets had complete Coverity analyses. The rest of the patchsets' analyses did not produce any results, for example, if the uploaded files were not C or C++ files, or if there was an internal analysis error. Of the analyzed patchsets, 1469 had passed the analysis without any issues, and 188 had failed. In the failed analyses, in total 1164 Coverity issues were flagged.

This means that on average, there were 0.702 issues per patchset with a completed analysis, and on the failed analysis patchsets (with at least one issue), there was a median of three

issues. Most reported issues originated in C code, some in C++ code. Notably, some (but few) patchsets got a large number of warnings, with the maximum number being 47 warnings. When looking over the type of flagged issues, it is evident (as found also during our user research) that there is mainly a small group of warnings that are triggered a lot of times. For example, warnings about unchecked return statements were produced a total of 249 times.

7.2 User engagement

In this section, we present different ways to measure user engagement in the feedback system, by derivation from implicit data of the interaction between users and various parts of the system (the feedback form, the game elements, and the Confluence space), and by adding complementary insights from the chat interviews. For all relevant engagement statistics, comparisons are also made between the gamification and control groups.

Notably, user engagement is difficult to measure for a number of reasons. First, we assume test group sizes (929 users: 464 gamification users and 465 control group users) based on the active number of Gerrit users during the three months prior to deployment, while the effective groups of relevant users may be considerably smaller. As explained in Section 3.4.2, the effective user groups can be said to be 302 users: 139 gamification users and 163 control group users. Second, three days of patchsets analyses were generated before the feedback system was deployed, during which users could not engage with the data, meaning that there at deployment time existed a "buffer" of previous entries to engage with. Third, any user could engage with any feedback form, making it difficult to say how many users (code owners, reviewers, and others) actually saw the feedback form, versus the ones that interacted with it. These factors lead to user engagement percentages being difficult to interpret, however, the more important measure is the comparison between the control group and the gamification group.

7.2.1 Feedback form

User engagement with the feedback form can be measured in multiple ways: the number of forms submitted, the number of active users (who submitted at least one form), the quality and completeness of submitted feedback, and more. Following subsections evaluate these metrics in order to give a nuanced picture of user engagement with the feedback system, looking at all relevant values separately for the gamification group and the control group.

Amount of collected feedback

The total number of submitted feedback forms was 66; 47 by the gamification group and 19 by the control group - i.e., gamification users left 147% more feedback than control group users. The feedback was given on 65 patchsets, which can be compared to the mentioned amount of patchsets possible to give feedback on: 1657. This means that 3.92% of the feedback forms saw any submit engagement. Because of previously discussed difficulties, it is not trivial to say whether this engagement percentage is high or low; more importantly, a difference can clearly be seen between the two groups.

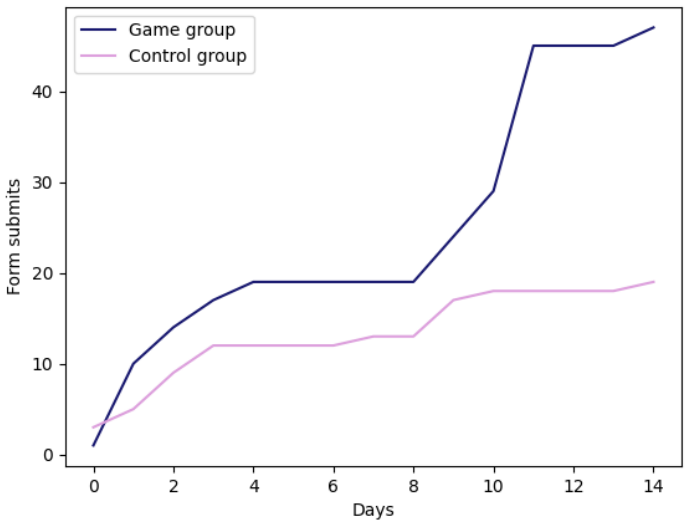


Figure 7.1: Accumulation of total submitted feedback forms over time, for both test groups.

Figure 7.1 shows how the submission of feedback forms grew over time both for the control group and the gamification group. This shows some interesting patterns. First, the long cease of submitted forms between days 4 and 8, and days 11 and 13, can be explained by the weekends and public holidays that occurred during these periods. The later sudden spike may be due to developers returning to work, and/or being influenced by the reminder email that was sent out after nine days of deployment (see Section 3.4.3 for details). Interestingly, this spike of engagement mainly originates from gamification users, perhaps since these users realized from the announcements that there were cookies to be acquired, or just got interested in the game features. Also, the spike on day eleven can be explained by the fact that the follow-up interviews took place at this time. Though both game users and control group users answered our questions, multiple of the asked game users (and no control users) seem to have taken a new interest in submitting feedback and thus did this after answering our questions.

Number of active users

For analyzing the engagement data and understanding user behavior, it is also important to check the number of active users in the two groups. Here, we define an active user as a user that submitted at least one feedback form (though there were other ways of being active, such as visiting the Confluence space).

In total, there were 30 active users, which is equal to 3.23% of our intended test group (and 9.93% of the approximated effective test group). Of the active users, 13 were control group users and 17 were gamification users. Thus, it seems like gamification did not encourage significantly more users to participate, but rather encouraged the participating users (those that already made the decision to submit feedback) to engage more. However, if looking at the approximated effective user groups (consisting of 139 gamification users and 163 control group users), the percentages of active users were 7.98% for the control group and 12.2% for

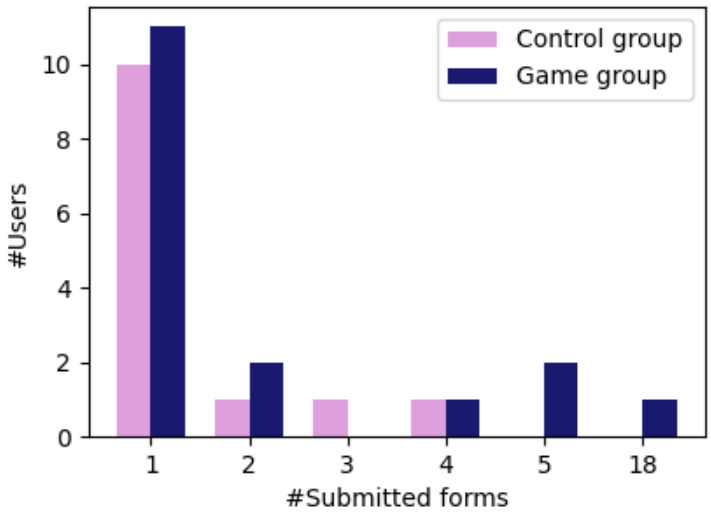


Figure 7.2: Number of forms that active users submitted in both user groups.

the gamification group, which indicates a difference that may not only have been due to individual user choices.

On average, an active control group user submitted 1.46 forms, while an active gamification user on average submitted 2.76 forms. However, these numbers may be heavily influenced by individual user behavior. For a more detailed view of how many users submitted different numbers of feedback forms, see Figure 7.2. The chart shows that almost all active users only submitted one form, while there are some clear outliers: notably, one gamification user submitted 18 feedback forms.

It is also interesting to note the time periods of individual user engagement behavior: when did the active users first submit feedback, and did individual users give feedback during brief periods of engagement or over a longer time? Of the active control group participants, the mean days of engagement (between the first and last submitted feedback form, where users with only one form are counted as having one day of engagement), was 2.15 days. For game users, the same number was 3.00 days. Figure 7.3 presents a more detailed view of active users' engagement over time. New users from the gamification group submitted their first feedback form throughout the entire test time (with the exception of the holiday in the middle). In the control group, new users engaged in the system mainly at the beginning of the test time and after the reminder email (day nine). As can be seen, more gamification users than control group users show prolonged, recurring engagement. Also, short and more intense engagements (leaving multiple forms in one day), mainly occurred in the gamification group, though also a few users in the control group submitted up to three forms on their first day.

As previously discussed, it is likely that many users in our original test group (as compared to the effective test groups) were not active in the feedback system since they do not use Coverity in their repositories. This was confirmed by the chat interview findings; multiple interviewees noted this as a reason for feeling that there was no purpose for them to give feedback. *P3* said that the little code he is responsible for is not relevant to Coverity, and *P10*

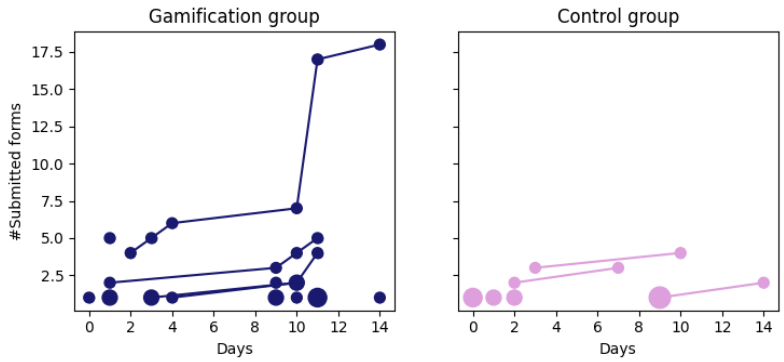


Figure 7.3: The feedback submit-history of all active users; an individual's statistics are connected by lines. Larger dots symbolize overlapping dots.

does not use Coverity much at all. *P5* said that without any failed Coverity analyses for his code, it did not feel meaningful to provide feedback, which shows that also Coverity-using members of our test group may have felt that there was nothing for them to give feedback on.

Engagement per repository

It is also interesting to compare the number of submitted feedback forms per repository, since this may indicate higher user engagement from users from certain code bases, for example where Coverity is especially appreciated or disliked. Note, however, that there may be multiple different reasons for discrepancies between repositories. First, it varies how many patchsets are uploaded and analyzed within a given time frame, and how probable it is to get Coverity issues – consequently, it varies how many patchsets and issues there are to give feedback on. Also, different repositories are maintained and visited in Gerrit by different amounts of active developers such as reviewers, thus changing the likelihood that one of these will provide feedback.

Of the total 66 submitted feedback forms, 38 repositories were involved. In Figure 7.4, the number of feedback forms per repository is shown, as well as the connections to the number of unique feedback givers on the repository. It is also shown how many patchsets were possible to give feedback on per repository (repositories with no submitted forms are not plotted). As shown, there was usually a relatively large amount of available feedback forms compared to the number of submitted forms. Though some repositories had only a small amount of available feedback forms, meaning that some active users might have given more feedback if there were more possibilities in their repositories. This may also be the case for users in repositories with a lot of available forms, as not all users will have insight into all changes in a whole repository.

The mean number of submitted feedback forms per repository, if only counting repositories with at least one submitted form, was 1.74. As also shown in Figure 7.4, most repositories only got one form submitted, while the most active repository had 8 submitted forms by 4 unique feedback givers. This shows that some repositories had a lot more user engagement, which did not only depend on individual developers.

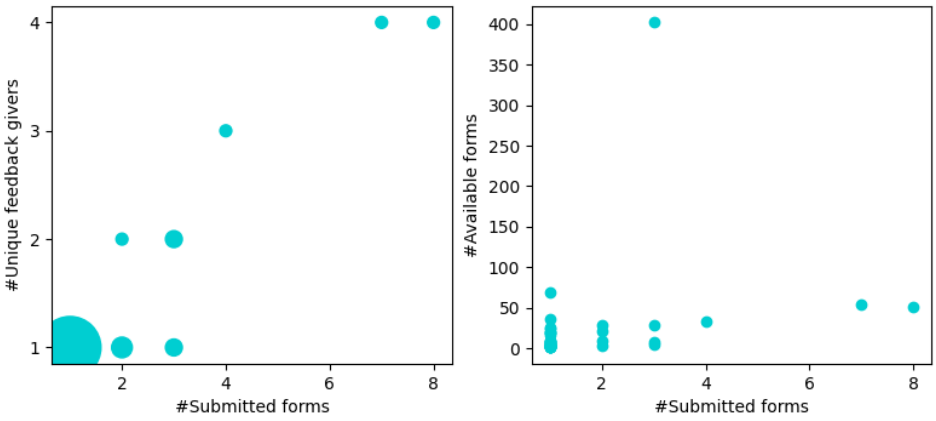


Figure 7.4: Number of submitted forms, unique feedback givers, and available forms, for all relevant repositories. Each dot represents a repository; overlapping dots are shown with larger dot sizes.

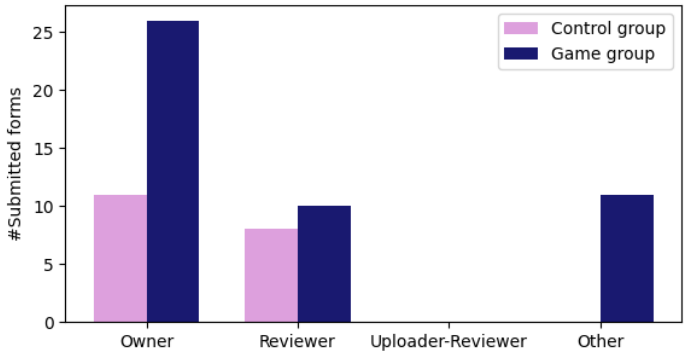


Figure 7.5: The number of forms submitted by different feedback-giver roles, for both test groups.

Types of feedback givers

Another way to examine user engagement behavior is to study which types of developers provided the most feedback. In our data, each feedback giver was assigned a role based on their Gerrit relationship with the patchset they gave feedback on. The roles were as follows: owner, reviewer, uploader-reviewer, and other, where "other" had none of the previously listed relations to the patchset. Figure 7.5 shows the division between feedback forms submitted by these roles and the differences between the gamification group and the control group. As seen, most feedback givers were code owners, which was expected since these are most likely to know enough about the analyzed code to easily give meaningful feedback. Still, a substantial amount of the feedback givers left feedback on patchsets they were only reviewers for. Also, notably, only gamification users left feedback on patchsets where they had no connection (at least not as owner, uploader, or reviewer). A theory is that these feedback givers went the extra mile to give feedback for the sake of the game features, perhaps when they had no more available forms in changes for which they were code owners or reviewers.

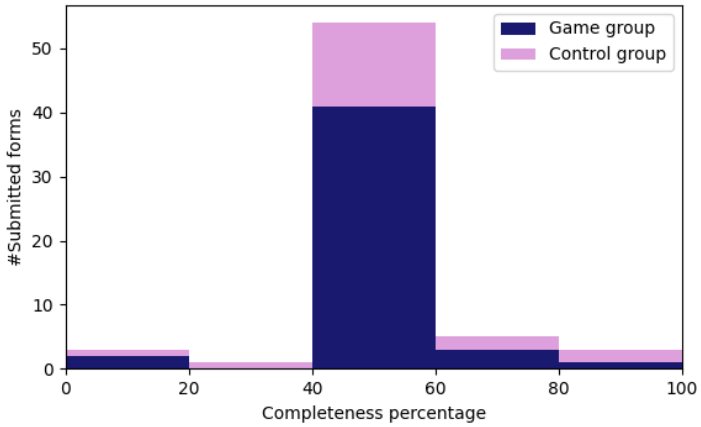


Figure 7.6: Distribution of total form completeness, for both test groups. Note that the bars are stacked.

Form completeness

In order to analyze user engagement in a more fine-grained manner, the completeness and quality of the collected feedback can be studied. Since feedback forms could be submitted even if they contained only one filled-in field, the completeness of the collected feedback forms may vary greatly. We define completeness here as the percentage of questions answered in the form - note however that different forms had different numbers of questions, based on the number of discovered Coverity issues.

The distribution of submission completeness is displayed in Figure 7.6. We found that forms were on average 51.5% complete; 53.89% for the control group and 50.53% for the gamification group, i.e., very similar values in the two groups. As seen in the figure, almost all forms filled in around 50%, and there is no obvious difference between gamification users and control group users.

Notably, there can be many different reasons for a form to be more or less filled in. A less filled-in form must not always mean lower user engagement per se: it is possible that users sometimes have nothing to say on different values. For example, we provided text boxes both for negative and positive things about Coverity, while a user perhaps only felt positively or negatively toward the analysis.

How much the forms are filled in likely also depends on the type of questions. The completeness percentages for multiple choice questions, vs full-text questions, are shown in Figure 7.7. This clearly shows, as also previously indicated by our user research (which measured motivation), that users are more easily engaged by multiple-choice questions. All these questions are usually answered in the forms, while full-text answers are much more rare.

Note also that the completeness of forms means slightly different things between forms where Coverity passed (and thus no issues existed in the form) and where Coverity failed since the latter will contain one or multiple issues to fill in except the general values. This may be one explanation for the outliers in our completeness scores. A theory for the overall completeness score being around 50% is that the forms regarding passed Coverity analyses most often were only given multiple choice answers, which in this case (almost always) constituted 50% of the form.

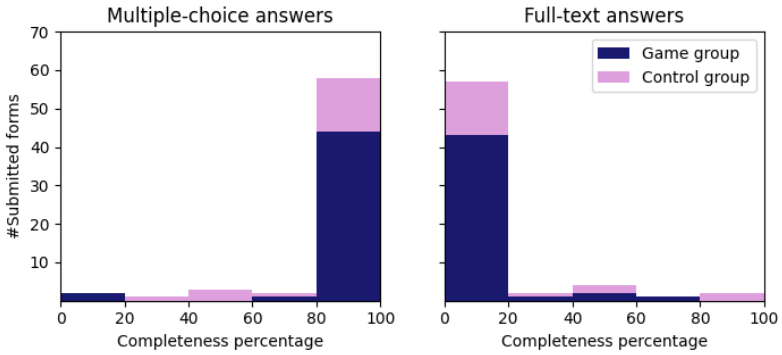


Figure 7.7: Distribution of completeness types for multiple choice and full-text answers, respectively, and for both test groups. Note that the bars are stacked.

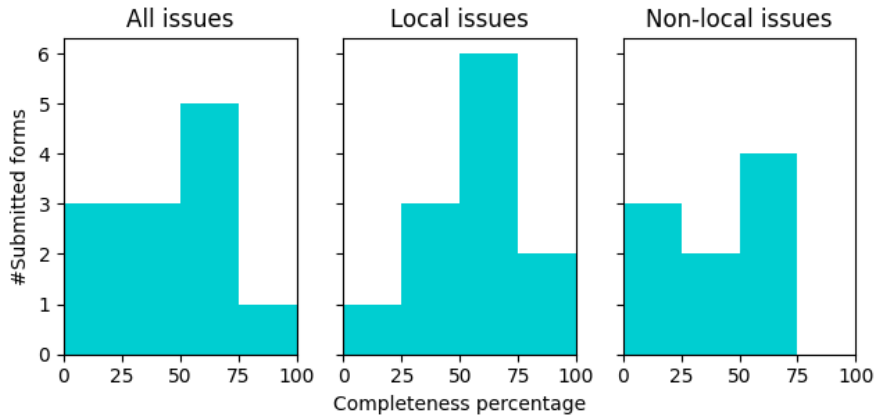


Figure 7.8: Distribution of issue completeness in submitted forms; in total, for local issues, and for non-local issues.

Issue feedback

Of the 1164 discovered issues, 48 unique issues received feedback at least once, i.e., 4.12%. In the submitted forms, there were in total 82 opportunities to give feedback on issues; 48 of these issues were in any way given feedback on, i.e., 58.5%, showing that users often had opinions on specific issues.

To get further insight, we examine the issue completeness of all submitted feedback forms, defined here as the percentage of filled-in form fields specifically for issues, vs. not-filled-in fields. In the 12 submitted forms that contained Coverity issues, the average issue completeness was 43.2%. When splitting between local issues and non-local issues, the issue completeness change to 57.0% and 34.2%, respectively. This is demonstrated in Figure 7.8. The data indicates that developers are more prone to filling in fields about local issues, i.e., issues introduced by the relevant patchset. This may support a finding made in our literature review (see Section 4.2), namely that developers prioritize issues that are relevant to their own code.

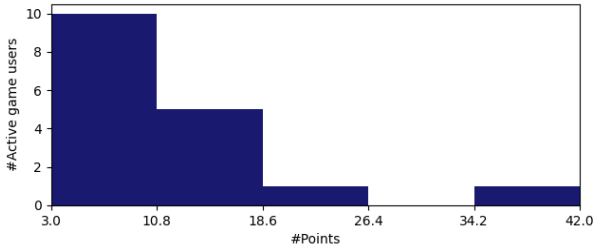


Figure 7.9: The distribution of acquired points among active gamification users.

7.2.2 Game elements

As a part of evaluating the efficacy of the gamification design, users' interaction with different game elements can be measured. This section looks into four of the main game elements in the feedback system: points, milestones (tightly coupled with extrinsic rewards), challenges, and the leaderboard, and evaluates statistics for these with regard to user engagement.

These statistics will mainly evaluate the engagement of the active gamification users, i.e., users that left feedback at least once. As previously mentioned, there were 17 active gamification users. Since this is a rather small group, statistics in this section should be observed with some caution, so as not to draw unjustified conclusions.

Similar to other results, the low engagement with gamification features may be partly due to users not knowing that these features existed, despite emails and announcements. This was noted by two participants in the chat interviews, who were gamification users and had provided feedback, but said they had not realized that the gamification features were accessible to them.

Points

During the whole deployment phase, a total of 177 points were collected by the active gamification users; 76.8% of these points came directly from giving feedback, the rest from achieving challenges. The average number of points for an active gamification user was 10.4 (for all gamification users in our test group, the number was 0.38 points, and for the effective test group 1.27 points), and a more detailed distribution is shown in Figure 7.9. As seen in the figure, most users got a relatively small amount of points, while a few users got significantly more.

On average, users received 3.85 points per submitted feedback form, i.e., a rather low number when considering the milestones (see below section for details). This may indicate that the game design was made too "difficult", as most feedback forms submitted by game users apparently were very simple; if these had rewarded more points (or if milestones had required fewer points), users would perhaps have been more inclined to leave more feedback.

It is however difficult to reason about the user engagement for points since users could not choose whether to get points or not; points were awarded automatically upon form submission.

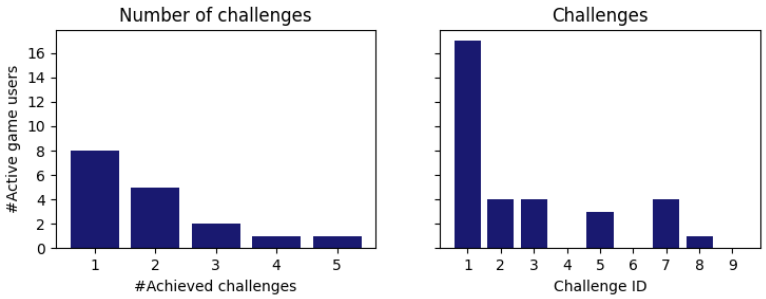


Figure 7.10: Achieved challenge count per active gamification user, and the distribution of which challenges were achieved by users.

Milestones

For all gamification users, the total number of reached milestones was 3, which is the same amount as cookie codes distributed. Of the distributed codes, none was actually (at the end of system deployment) used at the café as intended.

The average number of reached milestones per active gamification user was 0.18. One user reached milestone 1 (20 points), one (another) user reached milestone 2 (40 points), and zero users reached milestone 3 (65 points). This means that only two individual users interacted with the milestones.

The reason for the very low engagement with milestones, despite the extrinsic rewards, is perhaps the difficulty of reaching the milestones. As said earlier, users generally got 3.85 points per submitted feedback form, meaning that 5.19 average forms would be needed only to reach the first milestone; this is a lot to ask of the users, especially since they might not be familiar with that many Coverity analyses to give feedback on.

Challenges

In total for all gamification users, 33 challenges were achieved. The average of achieved challenges per active gamification user was 1.94 challenges/user – the full distribution is presented in Figure 7.10, which also shows the distribution of different achieved challenges. This does show some interaction with different challenges, though mainly by a few users. According to our user research, developers appreciate problem-solving. However, as noted in the system design description (Section 6.3.1), the challenges for the feedback system were difficult to make into problem-solving tasks; rather, they consisted of relatively simple goals for how to fill in the feedback form. This might be a possible reason for the low engagement with challenges.

It is worth noting that challenges, just as points, were rewarded automatically; thus, challenges may have been achieved without intention or awareness, and could therefore not truly be counted as user engagement (rather as user participation). For example, as can be seen in Figure 7.10, all active game users achieved Challenge 1, which was achieved when a user submitted their first feedback form. For most active users, this was also the only challenge they achieved.

Figure 7.11: Unique users per day visiting the Confluence space during the system deployment time.

Leaderboard

Notably, user engagement with the leaderboard competition element cannot be measured only by the implicit data, since engagement would only mean the act of watching the leaderboard count on the game dashboard or engaging with other developers in informal competitions. No evidence of user engagement with the leaderboard was given during the chat interviews; the leaderboard was only brought up by one participant (on his own accord) as an example of a game element that could be improved to increase motivation.

7.2.3 Confluence space

As described in Section 6.2.3, the Confluence space consisted of various different pages. Both overall space analytics and individual page view counts are available through Confluence. In all statistics presented below, our own page views and influence on the user count are excluded. Still, the numbers may in some cases be slightly higher than accurate since they are counted since page creation, and a small amount of Axis employees might have seen the Confluence space before the feedback system was deployed.

In total, the space front page (likely visited first by most users that visited the space) received 184 views from 97 unique users. Figure 7.11 shows how the views changed over time for the whole space. Notably, most new users visited when the system was first deployed, and after the reminder email was sent out. Except for these times, not many new users found the Confluence space.

Regarding the whole space, the average view count for the users that visited the space is 3.36 views per user. This indicates that there were not only problems with users not finding the space but also that the users that found it did not engage much with it.

Since view counts per default are anonymous in Confluence, we cannot tell how many views were from gamification users and not. However, the view count on the gamification dashboard page, which was only visible for gamification users, can be examined. In total, the dashboard received 57 views from 25 unique users; this indicates at least some engagement from a few users that visited the page more than once - the most active user who visited the game dashboard did so eight times. Notably, a smaller version of the game dashboard was also included on the Confluence space front page, thus gamification users could view their points without visiting the full game dashboard, which might have decreased the number of dash-

board views. In the chat interviews, one participant mentioned that he used the Confluence space to check his points.

The statistics pages in Confluence, in total, received 42 views. Of these, 20 views were on the Coverity analysis result dashboard and a table of found Coverity issues, and 22 were on one of the two dashboards regarding feedback results and feedback statistics, i.e., displays of the collected feedback and collection metadata. The numbers of unique viewers visiting the feedback results and statistics pages were very small: five and four respectively. This is unexpected, seeing that our user research indicated that users are more interested in seeing others' feedback on SCA tools than giving feedback themselves; in fact, the majority of developers in our questionnaire claimed they would like to be able to see others' feedback.

An explanation for the small view count may be that people did not know about the Confluence space. In the chat interviews, four interviewees (one gamification user, three control group users) said that they had not visited the Confluence page; *P7* and *P10* even said that they were not aware that the Confluence page existed at all, and the same was true for *P3* who only found out close to the end of the deployment time.

Another possible explanation is that developers may have felt that the task to read through the information or finding the dashboards took away too much from their work time, and/or was not relevant for them. After all, another finding in the user research was that developers highly value work efficiency.

7.2.4 Misuse/cheating

As indicated by previous gamification research (see Section 4.3), gamification solutions often run the risk of some users "cheating the system". However, we deem this risk low, since the user engagement with the game elements was so low; had there been more engagement, the risk of cheating and misuse of the system might have been higher. For example, users might have submitted meaningless feedback in order to get points. Now, at least when evaluating the full-text answers, all answers seem to be meaningful feedback.

The only indication of misuse of the system stems from one user, who, after having found out about the gamification features during the chat interviews, left a lot of simple, *Happy*, feedback on multiple repositories and quickly reached two milestones. This is however not necessarily "meaningless" feedback, and may only show a spike in user engagement.

7.3 User motivation

In this section, user motivation is examined, regarding the feedback form, interaction with the Confluence space, and interaction with the gamification features - partly to see if gamification increased user motivation. This is mainly done by analyzing the answers from the follow-up chat interviews.

7.3.1 Feedback form

When asked why they had left feedback or had not left feedback, and what motivated them to do either, the interviewees mentioned multiple different motivations. All participants had left feedback at least once.

Three of the participants wrote that the reason for them giving feedback was related to us having asked for feedback, i.e., they engaged with the feedback system in order to help out with our project.

Another commonly brought up reason for giving feedback was that the feedback may improve the Coverity system, or, as phrased by *P7*, their *"ways of working."* *P3* said that his motivation for giving feedback *"primarily was to help and make sure that what we are doing and how we are doing it becomes better."* This is a similar finding as from our previous research, i.e., that developers are motivated by work efficiency, and also are mainly motivated to give feedback when they think the feedback can leave to system improvement. When participants were asked if they felt there was any purpose for them to leave feedback, most participants responded positively for this same reason. *P2* felt that *"the purpose was to get better tools for code review,"* while *P1* noted that *"there has been a lot of discussions around how Coverity should be used over the past months, maybe the feedback can be used for this."*

Another, related motivation for participants to give feedback was for them to voice their opinions about Coverity, and/or since they felt strongly about the tool (and, for some, wanted to improve it). Two of the participants gave feedback because they felt that Coverity was important or useful. Two others saw flaws in the Coverity functionality, such as false positives, and wanted to report them.

When it comes to participants' motivation to not give feedback, *P7* found that it was difficult to take time to give feedback while being stressed. He also noted that he has *"recurring false positives that when you read them for the tenth time you lose the motivation to give feedback an eleventh time."*

7.3.2 Confluence

Regarding motivation for interacting with the Confluence space, *P2* said: *"[I] took a quick look at Confluence [and] it did not bring me any value. I was not interested in collecting points."* Also, though others seemed positive about the space, the rather low user engagement with the various pages indicates either that it was not well known or that users did not feel motivated to visit it - likely both.

7.3.3 Gamification

In terms of gamification and its effect on participants' motivation to give feedback, the interviewees that were also gamification users had mixed responses.

Some interviewees were positive, to different degrees, for example, *P8* said that he felt motivated by the game features and that he *"think[s] it is a good way to get good results."* *P6* found that gamification motivated better development: *"[It] encourages you to look through the Coverity reports even more thoroughly."* *P4* was motivated by the possibility of cookies through the gamification system – however, he was not part of the gamification group and therefore did not have access to this feature . Meanwhile, two participants said that their motivation was not connected to gamification, but rather to the prior mentioned reasons, like reporting flaws in the functionality.

7.4 User satisfaction

Similarly to user motivation, user satisfaction with the feedback form will mainly be evaluated via the chat interview findings. This was done partly to see if gamification increased user satisfaction.

7.4.1 Feedback form

Overall, the participants found that the feedback form was well-integrated into Gerrit, and did not disturb their workflow significantly; *P4* noted that the feedback form was designed well as *"it was so simple"* and *P9* that it was good to be able to provide quick feedback. Some explicitly expressed appreciation that the feedback form existed (should they want it), even if they did not find it relevant to them at the time; *P9* said he appreciated the ability to report likely errors with the Coverity analysis. These findings indicate that the system achieved the wish, found during user research, to not disrupt the development workflow. It also indicates that developers might want and appreciate the feedback form even though the user engagement numbers may make it seem like they do not.

Regarding possible improvements, *P7* thought that the feedback system worked okay, but that a better solution might have been to integrate the feedback functionality into the code view instead of as a separate element in the UI. This solution would have been more similar to MEAN [13] and its use of robot comments.

7.4.2 Confluence

The participants did not have many direct opinions about the Confluence space. Although, *P6* found the space interesting, specifically for seeing statistics about Coverity.

However, the Confluence space's display of the gamification features may not have been clear enough. As said by *P8*, who did not find the gamification entirely clear: *"I received points but did not see [anything] in Confluence, so [I was] a little confused if I was in [the] gamification [group] or not."* A similar sentiment was voiced by *P2*, who thought he was not a part of the gamification group (despite being so) as he had expected something different than what he saw on Confluence (this participant was however not interested in collecting points overall, which may explain a lack of motivation for trying to understand the game features).

7.4.3 Gamification

When asked about their opinions on the gamification features, participants *P6* and *P8* found them fun, while others were more skeptical. *P3* said that *"gamification is an interesting concept and I think when it is executed in the right way it can be both fun and motivating,"* but felt like our gamification system lacked some functionality, for example, a more comprehensive leaderboard and daily tasks to keep motivation up; similarly, *P1*, who was generally not a fan of gamification said: *"In the best case it is distracting, in the worst case it makes you feel like a pet being trained to walk prettily."*

The participant who had mentioned the lack of functionality (*P3*), especially the more extensive leaderboard, noted that it would have been more motivating for him if he could

see the entire leaderboard to get a more complete picture. This could still be anonymous but include the user's position in relation to everyone else. He added: *"[I]f I can see myself in relation to everybody else in [a list] then the competition element becomes more obvious and you get clearer visual feedback that you are climbing the ranks."* Furthermore, with such a leaderboard it would be possible to know in more detail how many points are needed to reach the first place and similarly, instead of only knowing the required points for reaching the next step on the ladder.

The game design was also not clear enough, according to some users. As noted by *P8*, he felt that it was not entirely clear what the gamification aspects involved. Another insight on the same topic was given by *P8*, who, on the question about what he thought on the Confluence space, said that he experienced his points seemingly being counted incorrectly. This was likely a problem with clarity in the gamification dashboard view, for example, the way in which challenge points were displayed. It seems like the game rules, or more specifically the existence and type of the game were not as self-explanatory as they were aimed to be.

7.5 Feedback content

In this section, we present an overview of the content results from the feedback form submissions. Although not directly connected to our research questions, this is important for discovering correlations in our results. For example, the overall satisfaction of the Coverity analysis may be representative of true satisfaction among users, or there may be a bias in which users gave feedback: did more satisfied users give the most feedback, or the other way around?

7.5.1 Satisfaction

In general, 70.0% of feedback forms received an overall satisfaction score of *Happy*, 16.7% *Neutral*, and 7.58% *Not happy*; the rest of the forms had no entry on this question. The satisfaction distribution is presented in Figure 7.12. Interestingly, there exists a quite large amount of *Neutral* feedback, though it has been noted in previous feedback research (see Section 4.2) that users without any opinions on the subject have less motivation to leave feedback. However, users may still have given positive/negative feedback on specific issues, even if they found themselves neutral to the analysis as a whole. As evaluated in Section 7.3, developers may also be motivated to give feedback by many other reasons than strong opinions, such as wanting to help us get results, or having an interest in the game elements. This may have led to some users submitting *Neutral* feedback rather than submitting no feedback at all.

In order to nuance the data, we also look at the satisfaction of individual users. Figure 7.13 shows how the satisfaction varied between different users and for feedback on specific repositories. The satisfaction scores have been calculated by assigning numbers (1 for *Happy*, -1 for *Not happy*), and calculating the mean number for each user and for each repository. The plots show that multiple users only submit positive feedback, and similarly that many repositories only get positive feedback (this is likely also correlated since single repositories often only got feedback from one or a few users).

Interestingly, as can also be seen in Figure 7.13, no users left only negative feedback forms (except those that only submitted one form). This may indicate that users who only feel

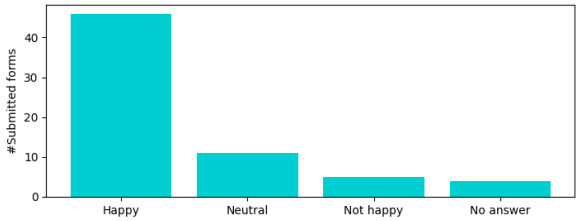


Figure 7.12: The entered satisfaction options in all forms.

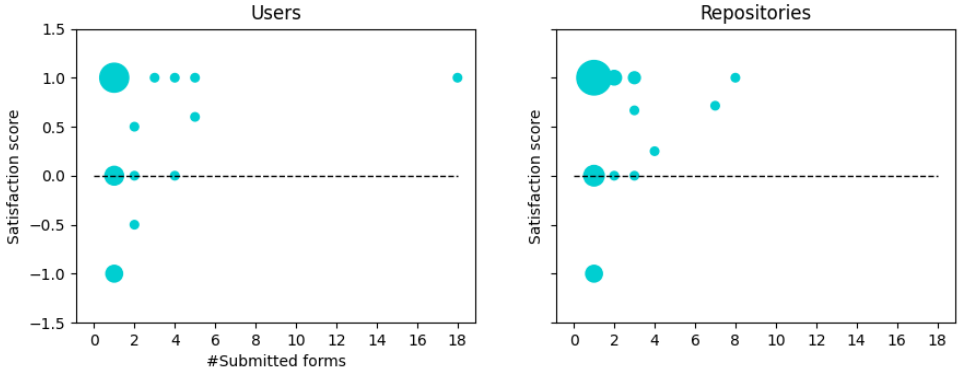


Figure 7.13: Satisfaction scores and number of submitted forms from individual users, and for repositories. The dot size represents overlapping values.

negatively toward Coverity (who exist, as we know from our user research) either do not have any analyses to leave feedback on, since they have disabled Coverity, or are not willing to engage with the feedback system. While our literature review (Section 4.2) mentioned that strong negative opinions may lead to users wanting to give feedback, our user research found that developers are more intent on improving SCA practices when they have an overall positive view of SCA; otherwise, they seem to prefer to not engage at all. This may explain the overall few occurrences of negative feedback. However, some negative feedback was given, by users only submitting one form; this may indicate that these users both dislike Coverity overall and are unwilling to engage more in feedback acquisition.

The range of Coverity satisfaction in the feedback also supports the user research finding that there are very different views on Coverity, positive and negative alike. Noteworthy, the satisfaction of different users may also depend on which type of analyses they got specifically during the deployment time, for example, if users appreciated passed analyses more than analyses that presented issues in the code. Users may (as indicated by our literature review) be less happy with the SCA tools when a large number of warnings are generated.

7.5.2 Issue tags

The issue tags entered by users in the multiple-choice section in the feedback form are shown in Figure 7.14. Note that this figure does not show overlap, i.e., two tags could have been given to the same issue, or to two different issues, and this difference would not reflect in the chart (as mentioned previously, issues received feedback in total 48 times, indicating a large overlap

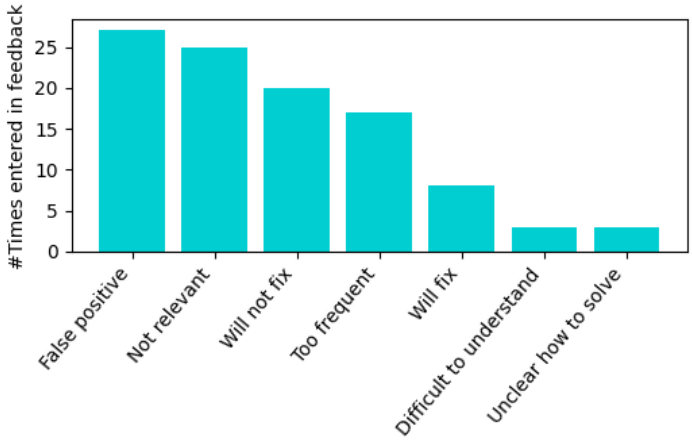


Figure 7.14: Tags given to Coverity issues in the submitted feedback.

in the figure). Notably, the only tag never given to an issue in the submitted feedback was the tag *Relevant*. The most common tags (*False positive*, *Not relevant*, etc.) indicate that users are not satisfied with the issues, even though the general Coverity feedback showed high satisfaction. Also, of all issues given feedback on, 25 issues were marked as *Not helpful*, while only 3 and 6 were marked as *Neutral* and *Helpful*, respectively. Again, this may indicate that users were predominately satisfied with patchsets that had no issues. It may also indicate that Coverity issues in general are not helpful - or, simply, that users are more engaged with giving negative feedback in the case of specific issues. Though, notably, all the issues rated as *Not helpful* stemmed from only 6 feedback forms, meaning that a few developers expressed their discontent with multiple issues at the same time - some of these also ranked other issues as *Helpful*, indicating that they may still appreciate the Coverity tool, even though they may be irritated by usability problems.

Chapter 8
Threats to validity

This section goes through possible threats to the validity of our thesis results, both regarding the user research results and the system evaluation results. The categorization of threats is taken from the list by Wohlin et al. [83]; we discuss threats to conclusion validity, internal validity, external validity, and construct validity.

8.1 Conclusion validity

Conclusion validity threats may hinder the ability to draw conclusions about whether the treatment (e.g. using gamification) actually led to the results (e.g. increased user engagement), i.e., they threaten the accuracy of the conclusions [83]. This section goes through a few such threats relevant to our thesis.

8.1.1 Short test period

The feedback system was deployed for a very limited amount of time, of which a substantial number of days were weekends and public holidays. This means that too little data may have been gathered to give complete results and arrive at confident conclusions; in the threat list by Wohlin et al. [83], these circumstances could be seen as a part of the *low statistical power* threat.

The short test period threat was mitigated by a few different factors: We used an as large as possible test group in order to collect as much feedback as possible, and we made it possible to also leave general Coverity feedback, not only feedback on Coverity-generated issues.

Notably, the short test period was specially raised as a concern for the feedback *content* results. During our user research, it became evident that Coverity issues, especially false positives, are relatively uncommon. This may mean that 14 days of testing could not provide enough data to show for example relations between encountered false positives and user

engagement. However, these types of results were not of major importance to our thesis conclusions.

8.1.2 Company culture misinterpretation

Another possible risk is that our description of the Axis company culture may not mirror the overall work culture. This is suspected since the results of the user research interviews give another picture than reported by our Axis supervisors and colleagues. According to them, Axis has a rather "playful" culture, where developers likely appreciate at least some elements of a playful user interface. Also, evidence has been shown that Axis developers often have a competitive drive and commonly like to compete, for example by the high participation in Advent of Code. Meanwhile, our user research shows a clear tendency *against* both a playful user interface and work-related competition.

These discrepancies may have different explanations. First, Axis is a large company and our interviews only included eight developers, though from some different departments and teams. Second, there may be a difference between the actual work culture and how employees view the work culture. Third, there may have been misunderstandings during the interviews, for example regarding the meaning of "competition" and "playfulness" - this could be classified as a construct validity threat but is mentioned here for simplicity.

Similarly to other user research risks, misinterpretation on this matter would also affect our feedback system evaluation, since the system design was based partly on our findings about the Axis company culture. In the threat list by Wohlin et al. [83], this may be categorized as a *reliability of measures* threat; because of lacking exactness in the measures, the reliability of the results may have been affected.

8.2 Internal validity

Internal validity threats occur when causality problems lead to unjustified cause-and-effect claims, for example, results depending on unknown factors instead of the measured factor [83]. In our thesis, both the user research results and system evaluation results have a few possible internal validity threats, especially since the design of the system evaluation was based on the user research results, making the results dependent on each other. This section lists and discusses these threats.

8.2.1 Selection bias

The result of any user study is affected by possible selection bias during the user sampling process; by Wohlin et al. [83], this threat is simply called *selection*. In our thesis, selection bias might have occurred in the interview participants sampling and possibly also in the questionnaire respondents sampling, if mainly developers interested in the subject of SCA tools and/or gamification chose to participate. Much of our interviewee sampling was done by asking Axis employees for colleagues who might be interested in participating. If this turned out to be a very biased sample, the user research results may have been misleading. This would also threaten the validity of our feedback system results and the analysis of these results, as many system design choices were made based on the user research results. For example, the

low engagement in gamification features may have indicated a biased and unsuitable game design rather than low user engagement for gamification as a whole.

However, we noted that the opinions of interview participants seemed to cover most of the spectrum of positive/neutral/negative regarding SCA tools. This indicates that possible selection bias may still not have had a detrimental effect on the distribution of our user research results.

It is also likely that mainly developers who wanted to help out with our thesis participated in our user research, and perhaps the same is true regarding the feedback acquisition. For example, in the questionnaire results, the Philanthropist player type may have been one of the main Hexad player types since mainly those with philanthropist tendencies wanted to answer the questionnaire. This is possibly also the case for the chat interview respondents: Mainly participants that had given feedback in our system also responded to our request for chat interviews, indicating perhaps that they were simply individuals who liked giving feedback in general.

8.2.2 Causal influences

One internal validity threat in the list by Wohlin et al. [83] is *ambiguity about the direction of causal influence*, which regards the difficulties of determining which factors influence each other. In our system evaluation, two possible threats on this theme regard the game element factors and the email reminder factor.

Simultaneous evaluation of game elements

Previous gamification experiments in literature often do not evaluate a game element in isolation, making it difficult to determine the effect of each element. This is the case also for our study, and thus the evaluations of the game elements given in Section 7.2.2 may not be entirely accurate: The user engagement for different game elements are likely dependent on each other and on the system design as a whole. However, this type of system evaluation was chosen in order to provide a gamification design as complete and suitable for the given context as possible. While it may have been possible to test each gamification element one by one, this may also have presented inaccurate results, for example since some game elements complement each other and since the novelty effect of the system would decline over time and affect the different game element evaluations.

Influencing users to engage

As explained in Section 3.4.3, users were originally informed about the feedback system via emails and Microsoft Teams announcements and a reminder email was sent out to all users after nine days of system deployment. Both the start of the system deployment and the time after the email reminder saw short spikes in higher user engagement. Since the email reached (in theory) as many control group users as gamification users, and was identical for all, we deem that this did not affect the comparison between user engagement between the two groups. However, it does affect the evaluation of overall user engagement for our feedback system. It may be argued that users gave feedback since they were prompted to, and in order to help us get results (which was admitted by some chat interview participants), rather than

intrinsic want to give feedback. This would threaten the validity of our user engagement scores.

A similar effect was seen during and after the chat interviews (which were conducted during system deployment because of thesis time constraints), where the interviewees were reminded about the system and the gamification features - some of these users immediately engaged more with the system.

8.2.3 Diffusion or imitation of group behavior

Wohlin et al. [83] lists the threat *diffusion or imitation of treatments*, which occurs when a participant in an experiment changes their behavior in accordance with the treatment applied to another test group. In our system evaluation, this may have occurred if the control group participants believed that they were gamification users. We know that this happened at least for one user, who in a chat interview said it was a motivation for him to give feedback because of the possibility of getting cookies. Both groups knew about the existence of the other group and the treatment they received since this was clarified in the announcements and reminders sent to all users. Some participants may not have checked, or not have understood how to check, which group they belonged to. This may have led to some control group users being motivated to give feedback in order to get game rewards or just to check whether they could find any game features or not. Similarly, as found in the chat interviews, multiple gamification participants that gave feedback had not realized that they were part of the gamification group and thus could not act by motivation connected to game rewards. These factors may have influenced our findings on the user engagement difference between the two groups.

8.2.4 Conscious altering of user behavior

Wohlin et al. [83] lists the internal validity threats *compensatory rivalry* and *resentful demoralization*, which occurs if participants in one test group think they received worse treatment than the other group. This may lead to participants changing behavior, either to consciously alter the results of the experiment or to perform worse because they become demotivated by not having the benefits of the other test group. In our system evaluation, this may have occurred for example if control group users felt demoralized by the realization that they could not get cookie codes or take part in the other game features. It may also have occurred if users in one group were strongly against gamification and wanted to disprove that it could lead to better results, either by engaging a lot in the system as a control group user, or not engaging as a gamification user. However, we have not seen any evidence of this occurring in practice.

8.3 External validity

External validity regards the validity of applying the study conclusions outside the context of the study, i.e., the generalizability of the results [83].

Overall, the user research, game design, and system design were done for an Axis-specific context. This may lead to our results not being generalizable outside of similar contexts, as gamification efficacy heavily depends on the culture in which it is examined. However,

our method, such as basing the design on previous user research, and how to choose design elements, can be generalized to other contexts.

Notably, since both feedback acquisition and gamification are recommended by research to be closely incorporated into users' (developers') workflow, our system design is also relatively context-specific for setups using Gerrit and other mentioned tools. Results may depend at least partially on how well these tools naturally interact with each other.

8.4 Construct validity

Construct validity is about whether the used constructs actually measure what they claim to measure, and construct validity threats include both experiment design threats and social threats [83]. In this section, a few relevant experiment design threats are presented.

8.4.1 Metric definition

Our metrics for the system evaluation (user engagement, user motivation, and user satisfaction) may not have been sufficiently defined; in the threat list by Wohlin et al. [83], this problem could be mapped to the threat *inadequate preoperational explication of constructs*. For example, we measured user engagement in many different ways which all gave different results. This situation arose partly since different definitions of these metrics exist in previous studies, and because we found that presenting just a single user engagement number may give a misleading picture of the situation. Because of this, we chose relatively loose definitions of our metrics and instead aimed to present more nuanced results.

It was also difficult to define and measure user motivation and satisfaction, as these must be examined in a qualitative way by looking at users' experiences. During the chat interviews, results may have been affected by participants interpreting our questions and metrics in different ways. For example, when evaluating user satisfaction, one question to the participants was *Did you appreciate the feedback system?*. Participants may have interpreted this differently depending on their interpretation of "appreciate", which may have led to participants omitting information or giving irrelevant information, making the user satisfaction metric less accurate.

8.4.2 Insufficient metrics

As mentioned, user engagement with our feedback system was measured in multiple different ways. However, we did not make a proper comparison regarding the *quality* of the collected feedback from the control group and the gamification group. This was partly omitted since feedback quality would depend on many factors and be difficult to measure objectively. Also, we did measure feedback quality in a basic way by looking at the completeness of the feedback forms and the percentages of multiple-choice versus full-text feedback, which did not show any evident differences between the groups - this may indicate that the feedback quality was similar for both groups. However, to measure feedback quality in a more accurate way, controls for "nonsense" feedback, and analysis of the full-text feedback, would have to be conducted.

A hypothesis, based on results from previous gamification studies in other contexts, is that gamification users would be willing to give large quantitative amounts of low-quality feedback in order to get game rewards. This would lead to our three metrics (user engagement, motivation, and satisfaction) not being sufficient for measuring the success of gamification for feedback collection. It may not be desirable to have the gamification group submit a lot of feedback, for the sake of the game, if the feedback quality decreased. In the threat list by Wohlin et al. [83], this may be categorized as the *restricted generalizability across constructs* threat, which appears when an experiment setup affects a studied construct (the amount of feedback) positively, but another, not studied construct (feedback quality) negatively, leading to an unintended side effect.

8.4.3 Questionnaire clarity

As in all studies using surveys, there is a risk that survey respondents misunderstand some questions, or the purpose of the questions, since they cannot ask for clarifications in the same way as during an interview. This may lead to inaccurate responses and flawed conclusions and is another threat that may be categorized as *inadequate preoperational explication of constructs* by Wohlin et al. [83].

The main sections in our questionnaire regarded feedback collection preferences (see Appendix C.3) and Hexad player types (see Appendix C.2). For the feedback preferences answers, the results quite closely mirrored previous research results, and the questions were also voluntary to answer so that participants who did not clearly understand a question could choose not to answer. Thus, the validity is likely high for the results of this questionnaire part. Still, we want to note that these questions were possibly difficult for participants to imagine, even with the provided examples, since they required participants to relate to imaginary situations in which they may never have been.

Regarding the Hexad player types results, the threat to validity might be higher. While the questions themselves were simply phrased, the context of the questions and the purpose of asking them may not have been obvious, leading respondents to be unsure of what to answer. This is theorized since the results show almost equal overall scores for different player types, which may indicate that many users gave rather similar scores to all questions, due either to lack of diligence, lack of interest, or lack of understanding. However, it is also possible that this was an accurate depiction of the player-type distribution for Axis developers.

Chapter 9

Discussion

This chapter includes a more thorough discussion of the results presented in this thesis – both the user research results and the system evaluation results. First, the possible reasons for the amount and type of provided feedback are discussed. Then, the effects of the gamification features, and possible reasons for these, are examined. This is followed by reflections on possible improvements that could be made to our method as well as alternative design ideas that have been discussed throughout this thesis work. In the end, a user engagement comparison is made toward the previous feedback-collecting systems Tricorder and MEAN.

9.1 Reasons for feedback results

This section examines, in more detail, the possible underlying reasons for the amount and type of collected feedback during the system evaluation experiment, i.e., why users provided or did not provide feedback.

As noted in the results (Chapter 7), there are many different ways to measure user engagement with our feedback system. For example, the number of submitted forms compared to the number of patchsets where forms could be submitted was relatively low: 65 of 1657 patchsets (3.92%) had at least one feedback form submitted. Meanwhile, the number of active users compared to our effective user groups was better (7.98% and 12.2% for the control and gamification groups respectively). Still, most of these active users only gave feedback once. An overall low user engagement percentage was in part expected since feedback acquisition systems, in general, see low user engagement (see Section 4.2).

The results may be due to multiple reasons. One reason, indicated by multiple chat interviews, was practical: The users did not have many interesting Coverity analyses to leave feedback on. Other possible reasons, both for users submitting feedback and not submitting feedback, are further discussed in the sections below.

9.1.1 System not noticed

A likely reason for the overall low user engagement is that the feedback system may not have been noticed by many users. While the system was announced both via email and in Microsoft Teams groups, these announcements may not have reached all intended users (many employees may have Teams-channel notifications muted and ignore non-personal emails). Some recipients may also only have glanced over the information and then judged it as not relevant to their work.

This was also made evident by many participants in our follow-up chat interviews, who said that they had not noticed some parts of the system, for example, the Confluence space or the game features. However, it is worth noting that all of these interviewees had given feedback at least once prior to being contacted for the interviews; to get a complete understanding of the situation, the perspectives of people who had not submitted any forms would be necessary. This was in part given by two users who were contacted but did not participate in full interviews - however, their reason for neither noticing nor engaging in the system was that they did not use Coverity in their work, and therefore the feedback form was not visible for them in Gerrit. One of these users still mentioned that they had noticed the announcement email and that there existed a Confluence space.

Another possible sign that users had not noticed the feedback system after the first announcement is the spike in engagement, also from new users, after the reminder email. However, this may also have occurred if users noticed the system but forgot about it, or did not feel motivated to give feedback until receiving the reminder.

Gamification not noticed

It may also be the case that specifically gamification users were not fully aware of what it meant for them to be a part of the gamification group. For example, users may have been unaware that they could get cookies from the café, even though this was noted in the announcements and made clear in the game dashboard and Confluence information pages. This may partly explain the low user engagement with the game elements.

Unintrusive design

The risk of the system being unnoticed was one of the reasons for placing the feedback form directly in Gerrit, which was supposed to make it difficult to overlook. Meanwhile, another goal of the system design was to make the system unintrusive, for example by having a collapsed feedback form. While the system achieved the goal of being unintrusive, according to most chat interview participants, this likely resulted in the form being relatively easy to glance over, which was also made evident during the pilot testing. However, the alternative design decision of having the feedback form expanded by default was disregarded during the design phase. While it may have prompted more feedback, leading to higher user engagement, it would also likely have been perceived as an annoying and intrusive feedback request and may have disrupted developers from completing their normal tasks. This is not recommended by previous feedback acquisition literature and also goes against our user research findings that developers value work efficiency and simple interfaces. Thus, this decision would likely have led to significantly worse user satisfaction results.

9.1.2 System not available

Another, though probably small, reason for the few submitted feedback forms may be the fact that the feedback system was unavailable during short periods of the test time. This was unrelated to the system design and due to issues with either Gerrit or Jenkins, which led to some Gerrit/Jenkins downtime. During this time, it was not possible to access the feedback forms, and/or feedback form generation was delayed. This may have delayed the in-stream of feedback or even prevented it altogether in some situations since a user may not have had the time or interest to revisit a patchset later for giving feedback - or developers may have gotten stressed by the situation and thus less inclined to take the time to give feedback.

There was also a related, likely more important, issue regarding how the system was deployed. After the announcement of the system, there was a delay of around an hour until the system deployment. Thus, developers may have read the announcement and visited the Confluence space and Gerrit, without finding the feedback form (as was also seen to be happening in the Confluence view count at the time). This may have led users to believe that the feedback system was not relevant to them, or led them to lose interest and then forget about the system. Both of these possibilities would likely have negatively affected the probability of these users giving feedback later.

9.1.3 No purpose to give feedback

One thing mentioned during one of the chat interviews was the lack of information about the project and its purpose. As said by *P3*: *"I feel that information around that this [the project] existed and why it existed was inadequate."* He mentions that the likelihood of him giving feedback would have been higher if those two points had been more clearly communicated. While this mainly indicates problems in reaching out with information, as discussed above and confirmed by multiple other chat interviewees that had not found the Confluence page, it also indicates that the purpose of the system is important to users.

This finding was also supported by other chat interviewees, who noted that they were motivated to give feedback that would improve the system and the way that Axis works with static analysis. Also, as found in our user research (both the questionnaire and the interviews), developers are mainly motivated to give feedback if they feel that the feedback can influence the system directly, or if they know the feedback can help other users. I.e., a purpose in the form of some sort of feedback loop is desired.

A feedback loop could have been implemented as a way to increase the sense of purpose for the users to give feedback, for example, by turning off certain warnings after they were marked as *Not helpful* in the feedback, as done for example in the MEAN [13] system. However, this was not possible in our system design. Instead, a couple of strategies were implemented in order to make the feedback feel helpful for users and for system improvement: All feedback was posted visibly to all users, in order to promote discussion and reflection on Coverity, and the feedback results were aimed to be used for future SCA tool decisions at Axis, such as licensing decisions.

However, these intended purposes for the feedback system may not have been obvious to users, though the information was stated on the Confluence information pages and stressed in the announcements. Or, users may have found that the system's feedback loop was too slow, or the possible SCA tool processes improvements too indirect for motivating giving

feedback.

9.1.4 Prioritization of work efficiency

As found during user research, Axis developers often highly value the efficiency of their work and feel intrinsically motivated by work tasks such as product improvement. This may be one reason for developers not shifting their focus from their work tasks (which would be the context in which they visit Gerrit) to open a feedback form and take the time to fill it out. Giving feedback and visiting the Confluence page may even have been seen as a possible disruption in the workflow, which both the literature on feedback acquisition and our own user research suggests can de-motivate users from giving feedback.

9.1.5 Philanthropy

Some users likely gave feedback mainly in order to help out with our thesis, as a service to us. This is in part indicated by the spike in feedback given after the reminder email (in which we noted that we needed more feedback for our results); while this could also be due to other reasons, multiple participants in the chat interviews gave answers in support of this hypothesis. This also seems to be in accordance with the finding that Philanthropist is the most common primary player type among Axis developers.

This philanthropy aspect of the Axis culture has also been noted during the rest of our thesis work when looking for interview participants, questionnaire answers, and others willing to help out during working hours.

9.1.6 Feedback giver user types

While the Hexad player type distribution of Axis developers has been examined and discussed in this thesis, feedback-giving preferences of different user groups have not been handled in the same detail. Some possible user categorizations in this context, described in Section 4.2.5, are *feedback antagonists*, *passive and stingy people*, *privacy fanatic and generous people*, and *privacy tolerant and socially ostentatious people*.

Likely, all these user groups exist in our user base. The presence of the first two (*feedback antagonists* and *passive and stingy people*), is partly seen in the questionnaire results, where multiple users showed no interest in giving feedback on SCA tools. For these user groups, being asked to provide feedback in the first place, and then reminded through our additional announcements, might have dissuaded them from providing feedback and even led to irritation and negative views of the system.

The other two user types (*privacy fanatic and generous people* and *privacy tolerant and socially ostentatious people*) were likely more present in our user research since their participation indicated a want to help out with our thesis. Probably, these people also stand for the majority of the provided feedback in the feedback system. It may be argued that most of the chat interviewees fall into either of these groups since they provided feedback both through the feedback system and the interviews.

9.2 Gamification efficacy

Overall, the percentages of active users in the gamification group compared to the control group (12.2% versus 7.98% respectively, if regarding the approximated effective user groups) indicate that more users were willing to engage with the feedback system if it had gamification features. However, since the number of active users was so small, this may still be due to individual preferences, and thus no definite conclusion can be drawn. Regarding the overall numbers of submitted forms, the results are similar: Clearly, gamification users provided more feedback, but the numbers may mainly be due to one single gamification user who engaged a lot with the system, creating an outlier that skewed most other results.

The user engagement for different game elements, such as the cookie code rewards, seems very small - most engagement may have come from automatic rewards to users who may have been more interested in giving feedback than in the gamification aspects. The chat interview findings conclude that while a few users found the gamification idea to be fun, multiple others (as already seen in our previous user research) were mainly motivated by work efficiency and results.

This section elaborates more on the possible reasons for these gamification results, mainly the reasons for not seeing more engagement with the game elements.

9.2.1 Game design specifics

The gamification results likely depend to a great extent on the specifics of our game design. While an attempt was made to make the game design optimal for the given context, various issues (implementation limitations, selection bias in the user research, etc.) may have impeded this mission. Thus, the results can perhaps not confidently be interpreted as showing the success or failure of gamification overall in this context. Though to some point clarified by the chat interviews and the analysis of our previous user research, it cannot be said if the low user engagement with our game elements was due to users not being motivated by gamification, or simply due to a sub-optimal game design - likely, both of these aspects mattered. This section evaluates some of the aspects of our game design that may have had a negative impact on the resulting user engagement, motivation, and satisfaction.

Game constants

A potential problem is how the constants in the game were set, specifically the milestone limits (i.e., how many points were needed to reach a milestone and get a cookie code reward), and how many points were awarded for giving feedback and reaching challenges. Initially, during implementation, the milestones and point rewards were set in order to make it very easy to reach the first milestone. This was intended to quickly get users interested in the program and to make the cookie code functionality evident (since it otherwise might not have gained attention). However, during pilot testing, testers were asked if they found it too easy to get rewards, and said that it was way too easy - they recommended that one would have to give feedback at least 10 times to get a reward. In the end, a middle ground was found, where we estimated that around 5 average feedback forms would have to be filled in to get the first cookie code. However, seeing that user engagement for feedback collection usually is very low, this is a lot to ask of a user.

The possibility to get many points also varied significantly between users. It was likely difficult for most users to acquire even the points needed for reaching the first milestone, as they first had to find multiple Coverity analysis results that they knew enough about to give feedback on. If users could not within a small time frame access this amount of relevant Coverity analyses, they may have found it too tedious to even try reaching the first milestone, and thus they got no positive incentive from the promise of extrinsic rewards. All in all, the game design turned out to be unfair in favor of a few developers. This was impossible to avoid entirely for our context but could have been mitigated by using more suitable game constants.

In retrospect, it would likely have been better for the gamification results to keep the more generous game constants. For example, very few people received any cookie codes. Had more cookies been given, we theorize that the word about this would have been spread, making other gamification users realize the opportunity. Now, if participants felt that it was too difficult to get a cookie code, that may also have impacted their engagement with the gamification system negatively and possibly even made them negatively inclined toward the system, thus also decreasing user satisfaction.

Lack of social game elements

Another choice in our game design, which might have been detrimental to its success, was to not include any teamwork, collaboration, or social elements. During the chat interviews, we asked the participants if the feedback system had been discussed in their teams; all interviewees answered that the topic had not been brought up at all, except for *P2* who had talked about the feedback system with colleagues whose code he was to review. This, together with the low user engagement with the game elements, indicates that there was no social team engagement with the system.

The anonymous leaderboard likely did not contribute much to social competitions, even though it in theory allowed users to compete informally with colleagues by comparing leaderboard positions. An implementation of a full leaderboard where the user could see their own position in relation to others (as suggested by one of the participants in the chat interviews), would have been a way to increase the social aspect of this game element.

Our user research indicates that developers would have appreciated social game elements; the most appreciated type of competition seemed to be team-based competitions, either as an intra-team competition or as a team competition against other teams. Also, collaboration generally seemed more appreciated than competition. A theory is that our game design would have been more appreciated and gained more traction if organized on a team level, for example, as a collaboration task with team challenges and/or as a simple competition between teams. Similar game solutions were discussed during our system design phase, but de-prioritized in favor of keeping the competition anonymous, and because concerns had been raised during our user research about different teams having different conditions, thus making competitions unfair. This design choice was also made based on the difficulty of mapping certain users to certain teams (which may not always be a many-to-one mapping), and because this would have obstructed the randomized group assignment of the control group and the gamification group.

9.2.2 No team organizers

One possible way to increase user engagement with the game elements would have been to assign a sort of organizer role to a teammate in each relevant team, as suggested by previous research (see Section 4.3.3). This organizer could serve as a motivator for making the other team members interested in the tool, thus drawing more active users, and at least making the system known to users. This could be specifically beneficial for our system since one of its main problems seems to be that developers did not know of its existence. Also, the idea of team organizers leverages the theory that social game elements would have positive effects.

Notably, having team organizers would not have to be a solution only for the gamification system, but could also have been beneficial for the non-game feedback system, since users are probably more likely to engage if encouraged by teammates than if encouraged only by emails to a mailing list.

During system design, using team organizers was discussed but disregarded due to the same reasons as social game elements were disregarded, as described above.

9.3 Possible method improvements

Due to this thesis project having many phases and a limited time frame, we have had to make some sub-optimal decisions regarding the method – we acknowledge that this may have affected the results.

A possible improvement to the user research method would have been to conduct the questionnaire and interviews sequentially one after the other. This would have allowed us to use the results from the first study, for example, the questionnaire results, as a basis for designing the next study, for example, interview questions. Findings, and the validity of the findings, could in this way be examined further; the results of the first study could be supported or debunked by the second study.

Regarding the evaluation of the feedback system, an alternative method (which may or may not have been an improvement) would be to test the system in two phases: first without gamification and then with gamification. This approach is recommended by some gamification studies [57] and has the advantage of using a larger test group since the user base is never divided into groups. We instead chose the parallel evaluation approach because of time constraints, and in order to avoid the risk of a "novelty effect", i.e., that the system seems more interesting and leads to higher engagement at the beginning of the deployment, which may affect results when testing different features in different phases. The parallel approach, with a control group, is also recommended by other gamification studies, such as Hamari et al. [33]. We also considered our test group to be large enough to still give accurate results when divided into two groups.

9.4 Alternative system design ideas

During the system design phase of this project, many different ideas were discussed for the final feedback system solution as well as for the game design. While many of these have already been mentioned and discussed previously, a few are evaluated further in this section.

One alternative feedback system design would have been to ask for feedback on specific Coverity warnings via robot comments in Gerrit, instead of using a general feedback form. This is how feedback was collected in the MEAN [13] system. This approach would have allowed developers to give feedback "closer" to the context where the issue appeared since the robot comments would appear in the actual code that triggered the warning. Another similar idea, with the same benefits, would have been to prompt feedback on warnings in the Coverity HTML files where the warnings were displayed to the user. These approaches were disregarded since we wanted to provide a general feedback form in order to get feedback also on passed Coverity analyses, and we did not want to use both robot comments and a general feedback form, since we wanted to minimize confusion and cognitive stress for users. The approach was also disregarded since we wanted the system to be as non-intrusive as possible; developers might have felt that the feedback prompts were more annoying if they happened frequently with visible robot comments directly in the code display at Gerrit.

Another design choice that was discussed was the possibility of including a Gerrit vote relating to the feedback system. If no feedback had been given on a change, this vote would automatically have shown -1 on the change; when feedback had been given by some user, the vote would automatically change to +1. An extreme solution would have been to not let changes be merged if not having a +1 vote (as is the case of code review votes and similar), which would have forced users to give feedback. Another option would be to have the vote only to add users' incentive to give feedback. However, all of these alternative solutions would have made the feedback system much more intrusive, and likely been very detrimental to developer satisfaction. There is also a risk that users would start giving nonsense feedback only in order to get the +1 vote. Likely, this approach would also not have been recommended or allowed by the Gerrit-responsible employees at Axis. Because of these issues, this design choice was never seriously considered, but we found it to be an interesting discussion of how user engagement could be increased on behalf of other system quality aspects.

Another design choice that was frequently discussed during the system design phase was whether the game elements should be presented in an opt-in and opt-out manner. This was often brought up as important by interview participants in our user research, though mainly regarding competition elements. Originally, we planned to make it possible for a gamification user to "leave" the gamification group and thus opt out of the game features, alternatively providing the possibility of opting out of specific game elements such as the leaderboard. An alternative was to use a default setting of not being in the leaderboard, and letting interested users actively choose to participate. Data on how many opted in and out would also have been useful for measuring the appreciation of different game elements. However, this was disregarded in favor of making the game design anonymous and allowing users to "opt out" simply by not using the system. If the system was to be deployed for much longer, we believe it would be preferable to allow the user to customize which game elements they wanted to participate in, seeing the vastly different opinions on gamification and specific game elements among users.

9.5 Comparison to MEAN and Tricorder

As mentioned in the introduction of this report (Chapter 1.5), two main inspirations for our feedback system were Google's Tricorder and the MEAN system. With MEAN's definition of

user engagement (number of clicks per generated robot comment, i.e., number of clicks per generated warning to give feedback on), they have engagement numbers of 2.9% for MEAN and 0.8% for Tricorder.

For our system, there are mainly two ways to calculate a similar user engagement percentage for comparison. First, we divide the number of submitted feedback forms by the total number of Coverity analyses that could be given feedback on. This gives 66/1657 = 3.98%. Second, we divide the number of times feedback was given on Coverity issues (warnings) by the total number of issues possible to give feedback on. This gives 48/1164 = 4.12%.

Both these numbers indicate a comparative user engagement success of our feedback system. However, due to differences in our system design compared to MEAN and Tricorder, a few factors make it more difficult to directly compare these numbers. Notably, we started collecting patchsets before it was possible for users to give feedback, meaning that some forms likely were ignored because users did not visit the old patchsets anymore. However, calculating user engagement by including only patchsets stored from deployment and onward might also give misleading results, if some feedback submits were on patchsets created before system deployment. Another reason for comparison difficulties is that more people could interact with the feedback form in our implementation than in MEAN's; for example, people who are not owners or reviewers.

Chapter 10
Conclusions

In this chapter, we present our conclusions to this thesis. First, we summarize our findings, focusing on conclusions surrounding our research questions, then present some ideas of interest for future work within this research area.

10.1 Summary of findings

This thesis answered five research questions regarding feedback on static analysis and the gamification possibilities of this type of feedback collection.

When examining RQ1 (*What does previous research say on the effectiveness of different gamification methods?*), we found that previous research shows that gamification may have some promise for increasing user engagement, although criticism has been directed toward the accuracy and validity of these findings. Gamification, in general, seems to have had the most effect on tasks that previously has seen low user engagement. Meanwhile, findings related to RQ2 (*What does previous research say on usability issues with static analysis tools, and on giving feedback on software like these tools?*) show that SCA tool engagement is often low due to multiple usability issues, such as false positives and workflow integration issues. Giving feedback on software also faces low user engagement due to things like poor timing of feedback requests and the effort perceived to be too high.

Our user research confirmed multiple literature review findings, which aimed to answer RQ3 (*From a developer's perspective, which aspects hinder interest and engagement with static analysis tools and with giving feedback on these tools?*). It was found that the perception and opinion of SCA tools vary significantly between individuals and teams due to repository-specific SCA problems. We saw similar reasons for low engagement as in the research, such as false positives and configuration problems.

As an answer to RQ4 (*In which ways can gamification be used to increase the engagement with giving feedback on static analysis tools?*), we developed a feedback system and gamification solution following findings from previous steps, notably: closely integrated into developers'

usual workflow, not intrusive, having minimal playful user interface elements, and with an anonymized leaderboard without stressing the competition element.

The feedback system results answered RQ5 (*When evaluated at Axis Communications, does gamification increase interest and engagement for giving feedback on static analysis tools?*). In general, it was found that while gamification did not particularly increase the number of participants, it did increase user engagement among active users; the gamification group was found to give 147% more feedback than the control group and have slightly prolonged engagement time with the system. It was also seen that gamification users interacted more with providing feedback on things unrelated to them (such as static analyses for code where they were neither the code owner nor a reviewer). These results, however, were not entirely supported by the follow-up interviews, where most participants expressed no particular enthusiasm for the gamification features.

10.2 Future work

Many possibilities exist for further research on the topic of feedback collection on SCA tools and the gamification of these.

Foremost, the same study as done in this thesis could be conducted again but with the results presented here used as a stepping stone, both for a better feedback system and a better gamification solution. Some suggested method improvements are listed in Section 9.3; for example, the experiment could run longer. Possible enhancements of the gamification solution are discussed in Chapter 9, e.g., using social game elements, pushing further to make the system known and visible (since it seems appreciated by the few users that noticed the game elements), and making the game elements more obvious for users. Since gamification seems promising for increasing user engagement in feedback collection, but our system seems not to have gotten the whole way, this could provide further insights into whether gamification, if done correctly, is the way to go for increasing the amount of collected feedback.

If a similar experiment is done in the same context at Axis, this may also evaluate how much the novelty effect of the gamification features mattered, if at all. Also, a more thorough study could consider each game element individually to give a more accurate view of how different game elements affected user engagement, motivation, and satisfaction.

Another topic on where gamification may be evaluated, and on which little research exists today, is to gamify directly in static analysis tools, for making the users utilize tools as well. While this is not the purpose of our study, the lack of such studies was noticed during our literature review.

References

[1] Brittany Johnson, Yoonki Song, Emerson Murphy-Hill, and Robert Bowdidge. "Why don't software developers use static analysis tools to find bugs?" In: *2013 35th International Conference on Software Engineering (ICSE)*. IEEE. 2013, pp. 672–681. DOI: 10.1109/ICSE.2013.6606613.

[2] Nasif Imtiaz, Akond Rahman, Effat Farhana, and Laurie Williams. "Challenges with responding to static analysis tool alerts." In: *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE. 2019, pp. 245–249. DOI: 10.1109/MSR.2019.00049.

[3] Malik Almaliki, Cornelius Ncube, and Raian Ali. "The design of adaptive acquisition of users feedback: An empirical study." In: *2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS)*. IEEE. 2014, pp. 1–12. DOI: 10.1109/RCIS.2014.6861076.

[4] Kurt Schneider. "Focusing spontaneous feedback to support system evolution." In: *2011 IEEE 19th International Requirements Engineering Conference*. IEEE. 2011, pp. 165–174. DOI: 10.1109/RE.2011.6051645.

[5] Melanie Stade, Farnaz Fotrousi, Norbert Seyff, and Oliver Albrecht. "Feedback gathering from an industrial point of view." In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE. 2017, pp. 71–79. DOI: 10.1109/RE.2017.9.

[6] Dora Dzvonyar, Stephan Krusche, Rana Alkadhi, and Bernd Bruegge. "Context-aware user feedback in continuous software evolution." In: *Proceedings of the International Workshop on Continuous Software Evolution and Delivery*. 2016, pp. 12–18. DOI: 10.1145/2896941.2896952.

[7] *Coverity*. URL: https://www.synopsys.com/software-integrity/security-testing/static-analysis-sast.html (visited on 06/08/2023).

[8] *Confluence*. URL: https://www.atlassian.com/software/confluence (visited on 06/06/2023).

[9] *Elastic*. URL: https://www.elastic.co (visited on 06/06/2023).

[10] *Gerrit code review*. URL: https://www.gerritcodereview.com (visited on 06/06/2023).

[11] *Gerrit code review for git*. URL: https://gerrit-review.googlesource.com/Documentation (visited on 06/06/2023).

[12] *Jenkins*. URL: https://www.jenkins.io (visited on 06/06/2023).

[13] Anton Ljungberg and David Åkerman. *Data-driven Program Analysis Deployment*. 2020. URL: https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=9021479&fileOId=9021484.

[14] Anton Ljungberg, David Åkerman, Emma Söderberg, Gustaf Lundh, Jon Sten, and Luke Church. "Case study on data-driven deployment of program analysis on an open tools stack." In: *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE. 2021, pp. 208–217. DOI: 10.1109/ICSE-SEIP52600.2021.00030.

[15] Caitlin Sadowski, Jeffrey Van Gogh, Ciera Jaspan, Emma Söderberg, and Collin Winter. "Tricorder: Building a program analysis ecosystem." In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 1. IEEE. 2015, pp. 598–608. DOI: 10.1109/ICSE.2015.76.

[16] Kevin Andersson and Mohammad Abo Al Anein. *Data-driven Deployment of Program Analysis Fixes*. 2021. URL: https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=9067212&fileOId=9067213.

[17] Caitlin Sadowski, Edward Aftandilian, Alex Eagle, Liam Miller-Cushon, and Ciera Jaspan. "Lessons from building static analysis tools at Google." In: *Communications of the ACM* 61.4 (2018), pp. 58–66. DOI: 10.1145/3188720.

[18] Carmine Vassallo, Sebastiano Panichella, Fabio Palomba, Sebastian Proksch, Andy Zaidman, and Harald C Gall. "Context is king: The developer perspective on the usage of static analysis tools." In: *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE. 2018, pp. 38–49. DOI: 10.1109/SANER.2018.8330195.

[19] *Visual Studio Code*. URL: https://code.visualstudio.com/ (visited on 06/11/2023).

[20] *Pylint*. URL: https://pypi.org/project/pylint/ (visited on 06/11/2023).

[21] *mypy*. URL: https://mypy-lang.org/ (visited on 06/11/2023).

[22] *clangd*. URL: https://clangd.llvm.org/ (visited on 06/11/2023).

[23] *PEP 8*. URL: https://peps.python.org/pep-0008/ (visited on 06/11/2023).

[24] *clang-tidy*. URL: https://clang.llvm.org/extra/clang-tidy/ (visited on 06/11/2023).

[25] *ESLint*. URL: https://eslint.org/ (visited on 06/11/2023).

[26] *Black*. URL: https://pypi.org/project/black/ (visited on 06/11/2023).

[27] *Sparse*. URL: https://sparse.docs.kernel.org/en/latest/ (visited on 06/11/2023).

[28] *Cppcheck*. URL: https://cppcheck.sourceforge.io/ (visited on 06/11/2023).

[29] *Axis History*. URL: https://www.axis.com/about-axis/history (visited on 05/30/2023).

[30] Alberto Mora, Daniel Riera, Carina Gonzalez, and Joan Arnedo-Moreno. "A literature review of gamification design frameworks." In: *2015 7th International Conference on Games and Virtual Worlds for Serious applications (VS-Games)*. IEEE. 2015, pp. 1–8. DOI: 10.1109/VS-GAMES.2015.7295760.

[31] Tommaso Dal Sasso, Andrea Mocci, Michele Lanza, and Ebrisa Mastrodicasa. "How to gamify software engineering." In: *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE. 2017, pp. 261–271. DOI: 10.1109/SANER.2017.7884627.

[32] Benedikt Morschheuser, Lobna Hassan, Karl Werder, and Juho Hamari. "How to design gamification? A method for engineering gamified software." In: *Information and Software Technology* 95 (2018), pp. 219–237. DOI: 10.1016/j.infsof.2017.10.015.

[33] Juho Hamari, Jonna Koivisto, and Harri Sarsa. "Does gamification work?–A literature review of empirical studies on gamification." In: *2014 47th Hawaii International Conference on System Cciences*. IEEE. 2014, pp. 3025–3034. DOI: 10.1109/HICSS.2014.377.

[34] Rodrigo Henrique Barbosa Monteiro, Maurício Ronny de Almeida Souza, Sandro Ronaldo Bezerra Oliveira, Carlos dos Santos Portela, and Cesar Elias de Cristo Lobato. "The diversity of gamification evaluation in the software engineering education and industry: trends, comparisons and gaps." In: *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE. 2021, pp. 154–164. DOI: 10.1109/ICSE-SEET52601.2021.00025.

[35] Gustavo F Tondello, Rina R Wehbe, Lisa Diamond, Marc Busch, Andrzej Marczewski, and Lennart E Nacke. "The gamification user types hexad scale." In: *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 229–243.

[36] Virginia Braun and Victoria Clarke. *Thematic analysis.* American Psychological Association, 2012.

[37] Gustavo F Tondello, Alberto Mora, Andrzej Marczewski, and Lennart E Nacke. "Empirical validation of the gamification user types hexad scale in English and Spanish." In: *International Journal of Human-Computer Studies* 127 (2019), pp. 95–111. ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2018.10.002.

[38] Richard Bartle. "Hearts, clubs, diamonds, spades: Players who suit MUDs." In: *Journal of MUD Research* 1.1 (1996).

[39] Nick Yee, Nicolas Ducheneaut, and Les Nelson. "Online gaming motivations scale: development and validation." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2012, pp. 2803–2806. DOI: 10.1145/2207676.2208681.

[40] Klaas-Jan Stol, Mario Schaarschmidt, and Shelly Goldblit. "Gamification in software engineering: The mediating role of developer engagement and job satisfaction." In: *Empirical Software Engineering* 27.2 (2022), p. 35. DOI: 10.1007/s10664-021-10062-w.

[41] Mohammad Tahaei, Kami Vaniea, Konstantin Beznosov, and Maria K Wolters. "Security notifications in static analysis tools: developers' attitudes, comprehension, and ability to act on them." In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021, pp. 1–17. DOI: 10.1145/3411764.3445616.

[42] Lisa Nguyen Quang Do, James R Wright, and Karim Ali. "Why do software developers use static analysis tools? A user-centered study of developer needs and motivations." In: *IEEE Transactions on Software Engineering* 48.3 (2020), pp. 835–847. DOI: 10.1109/TSE.2020.3004525.

[43] Maria Christakis and Christian Bird. "What developers want and need from program analysis: An empirical study." In: *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. 2016, pp. 332–343. DOI: 10.1145/2970276.2970347.

[44] Marcus Nachtigall, Michael Schlichtig, and Eric Bodden. "A large-scale study of usability criteria addressed by static analysis tools." In: *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2022, pp. 532–543. DOI: 10.1145/3533767.3534374.

[45] Tukaram Muske and Alexander Serebrenik. "Survey of approaches for postprocessing of static analysis alarms." In: *ACM Computing Surveys (CSUR)* 55.3 (2022), pp. 1–39. DOI: 10.1145/3494521.

[46] Farnaz Fotrousi, Samuel A Fricker, and Markus Fiedler. "The effect of requests for user feedback on Quality of Experience." In: *Software Quality Journal* 26 (2018), pp. 385–415. DOI: 10.1007/s11219-017-9373-7.

[47] Dennis Pagano and Walid Maalej. "User feedback in the sppstore: An empirical study." In: *2013 21st IEEE International Requirements Engineering Conference (RE)*. IEEE. 2013, pp. 125–134. DOI: 10.1109/RE.2013.6636712.

[48] Agnis Stibe and Harri Oinas-Kukkonen. "Using social influence for motivating customers to generate and share feedback." In: *Persuasive Technology: 9th International Conference, PERSUASIVE 2014, Padua, Italy, May 21-23, 2014. Proceedings 9*. Springer. 2014, pp. 224–235. DOI: 10.1007/978-3-319-07127-5_19.

[49] Joost Broekens, Alina Pommeranz, Pascal Wiggers, and Catholijn M Jonker. "Factors influencing user motivation for giving online preference feedback." In: *5th Multidisciplinary Workshop on Advances in Preference Handling (MPREF'10)*. 2010.

[50] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. "From game design elements to gamefulness: Defining 'gamification'." In: *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*. 2011, pp. 9–15. DOI: 10.1145/2181037.2181040.

[51] Carlos Futino Barreto and César França. "Gamification in software engineering: A literature review." In: *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE. 2021, pp. 105–108. DOI: 10.1109/CHASE52884.2021.00020.

[52] Kevin Werbach. "(Re)defining gamification: A process approach." In: *Persuasive Technology: 9th International Conference, PERSUASIVE 2014, Padua, Italy, May 21-23, 2014. Proceedings 9*. Springer. 2014, pp. 266–272. DOI: 10.1007/978-3-319-07127-5_23.

[53] Jane McGonigal. *Reality is broken: Why games make us better and how they can change the world.* Penguin, 2011.

[54] Titus Barik, Emerson Murphy-Hill, and Thomas Zimmermann. "A perspective on blending programming environments and games: Beyond points, badges, and leaderboards." In: *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE. 2016, pp. 134–142. DOI: 10.1109/VLHCC.2016.7739676.

[55] Gabriela Martins de Jesus, Fabiano Cutigi Ferrari, Daniel de Paula Porto, and Sandra Camargo Pinto Ferraz Fabbri. "Gamification in software testing: A characterization study." In: *Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 39–48. DOI: 10.1145/3266003.3266007.

[56] Stefan Stieglitz, Christoph Lattemann, Susanne Robra-Bissantz, Rüdiger Zarnekow, and Tobias Brockmann. *Gamification using game elements in serious contexts.* New York, NY, USA: Springer International Publishing, 2017.

[57] Andrés Francisco Aparicio, Francisco Luis Gutiérrez Vela, José Luis González Sánchez, and José Luis Isla Montes. "Analysis and application of gamification." In: *Proceedings of the 13th International Conference on Interacción Persona-Ordenador*. 2012, pp. 1–2. DOI: 10.1145/2379636.2379653.

[58] Michael Sailer, Jan Ulrich Hense, Sarah Katharina Mayr, and Heinz Mandl. "How gamification motivates: An experimental study of the effects of specific game design elements on psychological need satisfaction." In: *Computers in human behavior* 69 (2017), pp. 371–380. DOI: 10.1016/j.chb.2016.12.033.

[59] Fabian Groh. "Gamification: State of the art definition and utilization." In: *Institute of Media Informatics Ulm University* 39 (2012), p. 31.

[60] Lisa Nguyen Quang Do and Eric Bodden. "Gamifying static analysis." In: *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2018, pp. 714–718. DOI: 10.1145/3236024.3264830.

[61] Matthieu Foucault, Xavier Blanc, Jean-Rémy Falleri, and Margaret-Anne Storey. "Fostering good coding practices through individual feedback and gamification: An industrial case study." In: *Empirical Software Engineering* 24 (2019), pp. 3731–3754. DOI: 10.1007/s10664-019-09719-4.

[62] Ali Darejeh and Siti Salwah Salim. "Gamification solutions to enhance software user engagement–a systematic review." In: *International Journal of Human-Computer Interaction* 32.8 (2016), pp. 613–642. DOI: 10.1080/10447318.2016.1183330.

[63] Gabe Zichermann and Christopher Cunningham. *Gamification by design: Implementing game mechanics in web and mobile apps.* Sebastopol, CA, USA: O'Reilly Media, Inc., 2011.

[64] Karl M Kapp. *The gamification of learning and instruction: Game-based methods and strategies for training and education.* John Wiley & Sons, 2012.

[65] Rita Marques, Gonçalo Costa, Miguel Mira da Silva, Daniel Gonçalves, and Pedro Gonçalves. "A gamification solution for improving Scrum adoption." In: *Empirical Software Engineering* 25.4 (2020), pp. 2583–2629. DOI: 10.1007/s10664-020-09816-9.

[66] Will Snipes, Anil R Nair, and Emerson Murphy-Hill. "Experiences gamifying developer adoption of practices and tools." In: *Companion Proceedings of the 36th International Conference on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2014, pp. 105–114. DOI: 10.1145/2591062.2591171.

[67] Shawn Nikkila, Silvan Linn, Hari Sundaram, and Aisling Kelliher. "Playing in taskville: Designing a social game for the workplace." In: *CHI 2011 Workshop on Gamification: Using Game Design Elements in Non-Game Contexts*. 2011, pp. 1–4.

[68] Oki Priyadi, Insan Ramadhan, Dana Indra Sensuse, Ryan Randy Suryono, and Kautsarina. "Gamification in software development: Systematic literature review." In: *Lecture Notes on Data Engineering and Communications Technologies* 147 (2023), pp. 386–398. DOI: 10.1007/978-3-031-15191-0_37.

[69] Oscar Pedreira, Félix García, Nieves Brisaboa, and Mario Piattini. "Gamification in software engineering–A systematic mapping." In: *Information and Software Technology* 57 (2015), pp. 157–168. DOI: 10.1016/j.infsof.2014.08.007.

[70] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. "Steering user behavior with badges." In: *Proceedings of the 22nd International Conference on World Wide Web*. New York, NY, USA: Association for Computing Machinery, 2013, pp. 95–106. DOI: 10.1145/2488388.2488398.

[71] Silviya Dencheva, Christian R Prause, and Wolfgang Prinz. "Dynamic self-moderation in a corporate wiki to improve participation and contribution quality." In: *ECSCW 2011: Proceedings of the 12th European Conference on Computer Supported Cooperative Work, 24-28 September 2011, Aarhus Denmark*. Springer. 2011, pp. 1–20. DOI: 10.1007/978-0-85729-913-0_1.

[72] Daniel de Paula Porto, Gabriela Martins de Jesus, Fabiano Cutigi Ferrari, and Sandra Camargo Pinto Ferraz Fabbri. "Initiatives and challenges of using gamification in software engineering: A Systematic Mapping." In: *Journal of Systems and Software* 173.110870 (2021). ISSN: 0164-1212. DOI: 10.1016/j.jss.2020.110870.

[73] Katie Seaborn and Deborah I Fels. "Gamification in theory and action: A survey." In: *International Journal of Human-Computer Studies* 74 (2015), pp. 14–31. ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2014.09.006.

[74] Wael El Gammal, Nada Sherief, and Walid Abdelmoez. "User-based adaptive software development for gamified systems." In: *Proceedings of the 2020 3rd International Conference on Geoinformatics and Data Analysis*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 115–122. DOI: 10.1145/3397056.3397088.

[75] Gartner. *Gartner says by 2014, 80 percent of current gamified applications will fail to meet business objectives primarily due to poor design*. Press Release. Available at: https://cdn.pressebox.de/a/3467bdd08fcb2cf1/attachments/0535150.attachment/filename/2012GamficiationSpecialReport-November+27+EMEA.pdf. Nov. 2012.

[76] Andreas Lieberoth. "Shallow gamification: Testing psychological effects of framing an activity as a game." In: *Games and Culture* 10.3 (2015), pp. 229–248. DOI: 10.1177/1555412014559978.

[77] Wojciech Frącz and Jacek Dajda. "Developers' game: A preliminary study concerning a tool for automated developers assessment." In: *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. 2018, pp. 695–699. DOI: 10.1109/ICSME.2018.00079.

[78] Scott Grant and Buddy Betts. "Encouraging user behaviour with achievements: an empirical study." In: *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE. 2013, pp. 65–68. DOI: 10.1109/MSR.2013.6624007.

[79] Mathias Eggert and Melina Kriska. "Gamification for software development processes–relevant affordances and design principles". In: *Proceedings of the 55th Hawaii International Conference on System Sciences*. 2022.

[80] Malik Almaliki, Nan Jiang, Raian Ali, and Fabiano Dalpiaz. "Gamified culture-aware feedback acquisition." In: *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. IEEE. 2014, pp. 624–625. DOI: 10.1109/UCC.2014.99.

[81] Omar Azouz and Youssef Lefdaoui. "Gamification design frameworks: A systematic mapping study." In: *2018 6th International Conference on Multimedia Computing and Systems (ICMCS)*. IEEE. 2018, pp. 1–9. DOI: 10.1109/ICMCS.2018.8525900.

[82] Scott Nicholson. "A recipe for meaningful gamification." In: *Gamification in Education and Business* (2015), pp. 1–20. DOI: 10.1007/978-3-319-10208-5_1.

[83] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

Appendices

Appendix A
Interview consent form

This appendix shows the information given in the consent form signed by the authors and each interview participant during our semi-structured user research interviews. The actual consent forms also included space for signatures at the end of the form.

Consent to Take Part of Research

Project: Gamifying Security in Software Development (Thesis Project).

Research team: Emma Dahlbo (emmada@axis.com), Essie Lundmark (essiel@axis.com), Emma Söderberg (emma.soderberg@cs.lth.se), Lisa Eneroth (lisa.eneroth@axis.com).

Purpose: Interview about static code analysis tool use and experience, and gamification preferences.

Participant's role in the study: Participating in an interview for user research in the project.

Responsibility of the researchers
Before the interview begins, a member of the research team will have:

- Explained the purpose of the interview and the way in which any generated data will be handled.
- Explained what will happen during the interview.
- Explained the rights of the participant, including the right to withdraw at any time.

What will happen to the data
During the interview, your answers to the questions and discussions will be audio recorded. All data gathered from the interview will be kept confidential, and only made available to

members of the research team, or in case external quality assurance takes place, to assessors under the same confidentiality conditions.

Excerpts of this data and aggregated results will be used in publications and presentations, but under no circumstances will personally identifiable information be included in these publications.

Rights of the participant
I understand the purpose of the interview as explained to me by the researcher and agree that I can opt-out at any time, without consequence. If I choose to do this my data will be discarded. I understand that I can decline to answer any questions.

I agree that my answers to questions during the interview will be recorded and that this data will be handled in accordance with the section above.

Appendix B
Interview protocol

This appendix contains the interview protocol used during the more thorough eight interviews of our user research. Before the start of each interview, the interviewee was informed of the aim of the study and their rights, in accordance with the consent form presented in Appendix A. Then, questions were asked according to the subsections below. Since the interviews were semi-structured, questions and follow-up questions varied slightly between participants, based on their answers and the interviewer's previous knowledge of the participant.

Note that all questions were asked in Swedish since all participants were Swedish-speaking. Both the translations and the original Swedish questions are thus offered below.

B.1 Background questions

- Which team at Axis do you work at? / *Vilket team på Axis jobbar du i?*
- What's your role in the team? / *Vilken roll har du i ditt team?*
- How long have you been there? / *Hur länge har du varit där?*
- How long have you worked at Axis? / *Hur länge har du jobbat på Axis?*
- How long have you worked with development? / *Hur länge har du jobbat med utveckling?*

B.2 SCA tool questions

- Which tool do you use for code review? / *Vilket verktyg använder ni för code review?*
- What static code analysis tools do you use in your work? (E.g., Coverity.) / *Vilka statiska kodanalysverktyg använder du i ditt arbete?*

- Where in your workflow do you use SCA tools? [Examples: IDE, Git hooks, Gerrit, Jenkins] / *Var i ditt arbetsflöde/setup använder du dessa verktyg?*
- How much do you use these tools? / *Hur mycket använder du dessa verktyg?*
- How long is your experience with static code analysis tools? / *Hur lång erfarenhet har du med statiska kodanalysverktyg?*
- What are your motivations to use SCA tools? / *Vad motiverar dig till att använda statiska kodanalysverktyg?*
- Would you still use these SCA tools if it was not enforced by your team or the organization? / *Skulle du ha använt dessa statiska kodanalysverktyg även om det inte var obligatoriskt inom ditt team eller organisationen?*
- What is your general experience when working with SCA tools at Axis? What is positive, what is negative? / *Hur skulle du beskriva din upplevelse av att jobba med statiska kodanalysverktyg på Axis? Vad är positivt, vad är negativt?*
- Which of the SCA tools that you use do you find useful, or not useful, and in which ways? / *Vilka av de statiska kodanalysverktyg som du använder tycker du är användbara, och inte användbara, och på vilka sätt?*
- Are there many false positives? (I.e., the analyzer warns about something which is not actually erroneous?) / *Är det många falska positiva? (D.v.s. att ett verktyg varnar om något som egentligen inte är felaktigt.)*
- Is it possible to suppress warnings? Should it be? / *Är det möjligt att stänga av specifika varningar? Borde det vara?*
- What do you think of the warning messages? Do they give enough information to understand and fix the problem? Are they clearly written? / *Vad tycker du om varningsmeddelandena? Ger de tillräckligt med information för att förstå och fixa problemen? Är de tydligt skrivna?*
- How often do you find that a warning is difficult to understand? / *Hur ofta upplever du att en varning är svår att förstå?*
- What do you do if a warning is difficult to understand? [Examples: Leave for later, ask for help, ignore, research and fix, suppress the warning.] / *Vad gör du om du om en varning är svår att förstå?*
- What do you do if you suspect that a warning is a false positive? / *Vad gör du om du misstänker att en varning är en falsk positiv?*
- Would you like a way to offer feedback to the system, for example on which warnings are useful and not? / *Skulle du vilja ha ett sätt att ge feedback till systemet, t.ex. på vilka varningar som är användbara och inte?*
- Is there anything that do you wish existed but is not currently available in the tools? / *Finns det något du önskar fanns i verktygen som inte erbjuds i nuläget?*

B.3 Gamification questions

Before these questions, a short explanation was given on what gamification is and examples on how we might use it in our thesis, in order to give the participants some more context on the subject.

- Would you appreciate a playful user interface inside your SCA tool setup? (For example, cute monsters representing bugs.) / *Skulle du uppskatta ett 'playful user interface' i dina statiska kodanalysverktyg? (T.ex. söta små monster som representerar buggar.)*
- Would you appreciate a playful user interface if it was on a separate website but not visible in your development setup? / *Skulle du uppskatta ett 'playful user interface' på en separat gamification-hemsida, som inte påverkar din utvecklings-setup?*
- Would you appreciate playful competition at the workplace, for example, leaderboards on who fixed the most SCA-tool warnings last week? / *Uppskattar du att tävla mot andra på jobbet för skojs skull, t.ex. med en leaderboard som visar vem som fixade flest kodanalysvarningar förra veckan?*
- Have you experienced any previous playful competition at the workplace? How did you experience that? / *Har ni haft någon tidigare liknande tävling på jobbet? Hur upplevde du det?*
- Do you appreciate such competition within your own team? / *Uppskattar du sådana tävlingar inom ditt egna team?*
- Do you appreciate such competition on an organizational level? / Uppskattar du sådan tävling på företagsnivå?
- Do you appreciate such competition if it was on a team-vs-team level, where your team competes together? / *Uppskattar du sådan tävling team-mot-team, där ditt team tävlar tillsammans?*
- What do you think of sharing your data on a team-level, for example by showing leaderboards and individual scores for team members? / *Vad tycker du om att dela din data inom ditt eget team, t.ex. med leaderboards och individuella poäng synliga för andra teamarbetare?*
- What do you think of sharing your data on an organizational level, for example by showing leaderboards and individual scores for everyone at Axis? / *Vad tycker du om att dela din data med hela organisationen, t.ex. med leaderboards och individuella poäng synliga för alla på Axis?*
- What do you think of avatars as an anonymization technique, for example, if you were visible on a leaderboard but with a chosen name and a profile picture of an animal? / *Vad tycker du om avatarer som anonymiserings-teknik, t.ex. om du var synlig på en leaderboard, men med ett taget namn och en profilbild på ett djur?*
- What do you think of sharing your data, but keeping it anonymized, for example by only showing a mean score of everyone at Axis? / *Vad tycker du om att dela din data, men hålla den anonym, t.ex. bara genom att visa anonym statistik för alla på Axis?*

B.4 End question

- Is there anything you want to add, or ask about? / *Finns det något du vill tillägga, eller fråga om?*

Appendix C
Questionnaire

This appendix presents the questionnaire used in this thesis' user research phase. All questions, except the control variable questions, were graded on a 7-point Likert scale. At the beginning of each section, the information displayed to the user before the questions is also shown, in cursive.

Note that the questions in the Hexad player type section (Section C.2) are taken directly from the Hexad framework as described by Tondello et al. [37], and are depicted here only for readability.

C.1 Control variables

1. Age
    (a) 18–25
    (b) 26–30
    (c) 31–40
    (d) 41–50
    (e) 51–60
    (f) 60+
    (g) Prefer not to say
2. Sex
    (a) Male
    (b) Female
    (c) Other

(d) Prefer not to say

3. For how long have you worked in development?

   (a) 0–2 years
   (b) 3–5 years
   (c) 6–9 years
   (d) 10–20 years
   (e) 20+ years

C.2 Hexad player types

Below are 24 questions that examine what "player type" you most resemble, according to the Hexad framework by Marczewski. (Andrzej Marczewski. 2015. "User Types". In Even Ninja Monkeys Like to Play: Gamification, Game Thinking & Motivational Design. CreateSpace Independent Publishing Platform, 69-84.)

4. I like helping others to orient themselves in new situations.
5. Interacting with others is important to me.
6. The wellbeing of others is important to me.
7. Being independent is important to me.
8. I like being part of a team.
9. I like overcoming obstacles.
10. I like sharing my knowledge.
11. It is important to me to follow my own path.
12. It makes me happy if I am able to help others.
13. It is important to me to feel like I am part of a community.
14. Rewards are a great way to motivate me.
15. I often let my curiosity guide me.
16. Opportunities for self expression are important to me.
17. I enjoy group activities.
18. I enjoy emerging victorious out of difficult circumstances.
19. I like to provoke.
20. It is important to me to continuously improve my skills.

21. I dislike following rules.
22. I like mastering difficult tasks.
23. I like competitions where a price can be won.
24. If the reward is sufficient I will put in the effort.
25. I see myself as a rebel.
26. Return of investment is important to me.
27. I like to question the status quo.

If you want to receive the results of this test via email, you can enter your email address below. This means that your results from the entire survey will be linked to your email address non-anonymously – however, it will only be used for the purpose of sending you your results.

C.3 Leaving feedback on static code analysis tools

28. I would like to leave feedback on static code analysis warnings.
29. I would like to see others' feedback on static code analysis warnings.
30. I am motivated to leave multiple-choice feedback, for example to mark a warning as "false positive"
31. I am motivated to leave text feedback, for example describe why a warning is a false positive.
32. I feel more motivated to leave feedback if it can help other users.
33. I feel more motivated to leave feedback if it can directly improve the system.
34. I feel more motivated to leave feedback if there is no other feedback on the topic. E.g., voting on a warning to be marked as a false positive if there are no previous votes.
35. I feel more motivated to leave feedback if there is already previous feedback that is similar to my own opinion. E.g., voting on a warning to be marked as a false positive if previous voters are of a similar opinion.
36. I feel more motivated to leave feedback if there is already previous feedback that is different from my own opinion. E.g., voting on a warning to be marked as a false positive if previous voters are of a different opinion.

C.4 GDPR

37. I allow for my data to be stored for the duration of this master thesis, in compliance with GDPR.

Appendix D
Chat interview protocol

This appendix presents the chat interview protocol. The participants were contacted via Microsoft Teams and sent both a short informed consent message and some questions, as presented in the sections below.

Note that all questions were written in Swedish since all participants were Swedish-speaking. Both the translations and the original Swedish questions are thus offered below.

D.1 Informed consent

If you want, we would appreciate you answering a few questions here via Teams about your experience with our ongoing feedback collection in Gerrit (for our master's thesis). The answers would, in that case, be saved (confidentially), and used in our report and presentation, possibly as direct quotes, but not connect to your identity. / *Om du vill får du gärna svara på lite frågor här via Teams om dina erfarenheter. Svaren skulle i så fall sparas (konfidentiellt), och användas i vår rapport och presentation, kanske som direkta citat, men inte kopplas till din identitet.*

D.2 Questions

1. Did you leave feedback at any point via the form in Gerrit? Why/why not? / *Lämnade du feedback någon gång via formuläret i Gerrit? Varför/varför inte?*

2. Did you feel there was any purpose for you to give feedback? / *Kände du att det fanns något syfte för dig att ge feedback?*

3. What influenced your motivation to give/not give feedback? / *Vad påverkade din motivation till att ge/inte ge feedback?*

4. Did you appreciate the feedback system? Did you find it disruptive in Gerrit? Why/why not? / *Uppskattade du feedback-systemet? Upplevde du det som störande i Gerrit? Varför/varför inte?*

5. Did you visit the Confluence space? If yes, what did you think of it? If no, why not? / *Besökte du Confluence-sidan? Om ja, vad tyckte du om den? Om nej, varför inte?*

6. Do you know how your team interacted with our project? Was it mentioned in the team? / *Vet du hur ditt team interagerade kring vårt projekt? Nämndes det i teamet?*

7. Do you have any other thoughts or opinions? / *Har du några andra tankar eller åsikter?*

Were you one of the users who had access to gamification features? If so, we have a few more questions: / *Tillhörde du en av de användare som hade tillgång till gamification features? Om ja, så har vi några fler frågor:*

8. Did you notice any of the gamification features? How? / *Märkte du något av gamification-funktionerna? Hur?*

9. What did you think of the gamification features? Did you like/dislike them? / *Vad tyckte du om gamification-funktionerna? Gillade/ogillade du dem?*

10. Did you feel that the gamification features motivated you to give feedback? Why/why not? / *Upplevde du att gamification-funktionerna motiverade dig att ge feedback? Varför/varför inte?*

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2023-06-19

EXAMENSARBETE Gamifying User Feedback Collection on Static Analysis Tools
STUDENTER Emma Dahlbo, Essie Lundmark
HANDLEDARE Emma Söderberg (LTH), Lisa Eneroth (Axis)
EXAMINATOR Martin Höst (LTH)

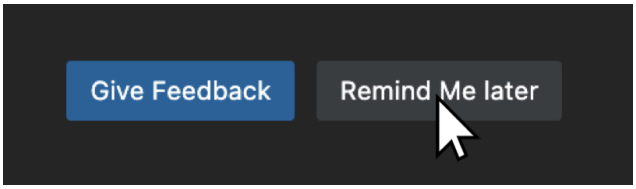
Samla feedback på statisk kodanalys - kan det göras engagerande?

POPULÄRVETENSKAPLIG SAMMANFATTNING **Emma Dahlbo, Essie Lundmark**

Det är allmänt känt att det är svårt att få folk att ge feedback. Vi har utvecklat ett feedbacksystem med spelelement för att maximera mängden feedback på statiska kodanalysverktyg. Under arbetets gång har vi upptäckt att mjukvaruutvecklare har många starka åsikter, positiva som negativa, om både statisk kodanalys och gamifiering.

Har du nyligen fått en uppmaning att lämna feedback på ett program, till exempel i ett programmeringsverktyg? Allt som oftast kommer det upp notiser och popups på hemsidor och appar som frågar vad du tycker om verktyget. Hur ofta klickar du ner dem? Skulle du överväga att *inte* klicka ner dem om du kunde få någon sorts belöning för att ge feedback – eller om det till och med framstod som ett spel? Vårt examensarbete visar att så kallad gamifiering av feedbackinsamling, alltså att lägga till spelelement såsom poäng till feedbacksystemet, kan öka mängden feedback som ges från mjukvaruutvecklare.

Vi har undersökt feedbackinsamling i kontexten av statiska kodanalysverktyg, det vill säga verktyg som letar efter buggar i programvarukod. Sådana verktyg är värdefulla för att få bättre kvalitet på koden, men ogillas ofta av mjukvaruutvecklare till den grad att de helt enkelt stängs av – och om verktygen inte används spelar det ingen roll hur bra de egentligen är. För att förstå varför det är så här, och hur det kan åtgärdas, kan man samla in feedback om statisk kodanalys – men mjukvaruutvecklare gillar inte att ge feedback heller. Vi har därför undersökt hur utvecklare kan motiveras bättre, och därefter utvecklat ett system för att maximera mängden insamlad feedback. Systemet möjliggör för utvecklare att ge återkoppling på det statiska kodanalysverktyget Coverity, varje gång Coverity publicerar en analys i kodgranskningsprogrammet Gerrit. Vi gjorde en "vanlig" version av systemet och en version där utvecklarna också kunde samla poäng, tackla utmaningar, och vinna kak-kuponger – alltså en gamifierad ("spelifierad") version.



Vårt system rullades ut till cirka 900 utvecklare på teknikföretaget Axis. Resultaten visar att utvecklarna som använde den gamifierade versionen gav 147% mer feedback. Samtidigt finns det väldigt skilda, och starka, åsikter om att lägga till spelfunktioner på det här sättet. I våra uppföljande intervjuer sade en utvecklare, angående gamifiering i allmänhet: *"I bästa fall är det distraherande, i värsta fall får det en att känna sig som ett husdjur som tränas att gå fint."* För att uppnå bästa möjliga resultat måste sådana åsikter också tas hänsyn till i designen av feedbacksystem. Vårt exjobb bidrar med en utförlig metod om hur gamifiering kan anpassas och utvecklas efter en specifik kontext och användarbas, både för att motivera utvecklare att ge feedback och för att göra dem så nöjda som möjligt med feedbacksystemet.

Nästa gång du får en notis som ber dig att lämna feedback, ägna en tanke åt vad som får dig att klicka "OK" eller att bara stänga ner notisen. Ett helt forskningsfält ägnar sig åt att försöka få dig att göra det förstnämnda.