Comparing prioritization and visualization of technical debt and security debt

A Design Science Research study

Sandra Liabø

Thesis submitted for the degree of
Master in Informatics: Programming and System Architecture
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Autumn 2022

Comparing prioritization and visualization of technical debt and security debt

A Design Science Research study

Sandra Liabø

Abstract

Context
Historical data from issue tracking systems provides valuable information about how technical debt has been prioritized in the past. These insights should be presented in a good way to support software practitioners in prioritizing the repayment of future debt.

Objective
In this study I focus on the difference between technical debt and security debt. I investigate the fixing rate of the debt to understand which issues are fixed in higher quantities. Further, I look into the lead time of the various issues to find out after how much time they are fixed. Finally, these insights are used as a starting point for creating an artefact to support the planning and prioritization of technical debt and security debt.

Method
A Design Science Research was conducted. I analyzed a data set of 10970 technical debt issues to find out which issues benefited the most in terms of repayment and time required to solve the issue. These findings together with requirements proposed by company stakeholders were the base from where I developed an artefact: a dashboard consisting of four different visualizations of historical data from Jira. The artefact has been evaluated through six interviews within two iterations.

Result
The results reveal that security debt has a significant higher fixing rate than technical debt. The calculation of lead time shows that technical debt is often fixed faster than security debt. Finally, an artefact has been provided as a start-

ing point for supporting the planning and prioritization of technical debt and security debt. The results reveal the visualization's degree of understandability, usefulness, and suggested improvements for further research.

Conclusion
The results from the two first research questions offers context knowledge for the rest of the thesis. In addition, they provide interesting empirical findings for research purposes. The findings from the last research question proves that the artefact offers a initial new way of approaching the prioritization of the debt by providing insights into the differences between security debt and the rest of technical debt, showing after how much time the debt have been fixed, as well as highlighting the most important unaddressed issues.

Acknowledgements

This master's thesis marks the end of five educational years at the University of Oslo. The work on this assignment has been exciting and fun, but also challenging alongside family life. Working on a master's thesis can sometimes seem lonely, but I have benefited greatly from wonderful people around me. There are many who deserve to be thanked for the fact that this thesis has turned into a finished product.

To my supervisor Antonio Martini (University of Oslo) and co-supervisor Daniela Soares Cruzes (Norwegian University of Science and Technology), thank you very much for all the feedback you have given me during the work on this master's thesis. You have challenged me on my thoughts, but also supported me throughout the work on this thesis. To both of you - thank you very much!

I want to thank the company stakeholders involved in our meetings. Your thoughts, input and ideas have contributed significantly to my learning during the work on this assignment! Additionally, I want to thank all of the participants for their contributions to making this study possible.

I want to thank my patient family, without their considerations it would have been difficult to complete my studies. Lastly, I want to thank all of my friends for supporting me in writing this thesis.

This marks the end of my education at the University of Oslo. It has been a pleasure!

Sandra Liabø
November 2022

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The pace of software delivery is increasing and technology is changing rapidly (Kruchten, Nord, Ozkaya, & Falessi, 2013). The highly competitive software market forces companies to work in tight schedules to be able to release software to customers faster (Yli-Huumo, Maglyas, & Smolander, 2016). As the emphasis on delivering functionality quickly is increasing, aspects such as design, good programming practices, and test coverage is often less focused on (Codabux & Williams, 2013). Because of this, these short-term aspects usually result in an increase in technical debt (Lindgren, Wall, Land, & Norström, 2008).

Technical debt is a popular metaphor in software engineering that has received significant attention in the recent years (Ciolkowski, Lenarduzzi, & Martini, 2021). The metaphor is used for technical trade offs that can be beneficial in short term but may hurt the health of the software system in the long term (Z. Li, Avgeriou, & Liang, 2015). Both in the research and industrial community, the metaphor has been further developed to contain different types of technical debt (Rindell & Holvitie, 2019).

As technical debt has become more mature as a concept, the term security debt has attracted more attention (Martinez et al., 2021). Although investigations that relates to security debt is recent (Ahmadjee & Bahsoon, 2019), it is a trend with ongoing research and new approaches (Martinez et al., 2021). The concept of security debt is derived from the technical debt metaphor (Martinez et al., 2021). At the TechDebt conference in 2021 it was found that the impact technical debt has on security is prominent and Ciolkowski et al., 2021 thus argues that such a relationship needs further investigation.

Both technical debt and security debt must be identified and managed in order to allow informed decisions on whether to take on the debt or not (Rindell & Holvitie, 2019). While the debt cannot be totally eliminated in practice, it needs to be managed to be kept at an acceptable level (Z. Li et al., 2015). To make the best decision about which debt items that should be repaid first and which items that can be delayed until later releases, the prioritization of technical debt and security debt is crucial (Alfayez, Alwehaibi, Winn, Venson, & Boehm, 2020).

In order to manage the debt one must know of the debt, and in order to know of the debt one must track it. If the issues are not tracked they will remain invisible (Martini, Besker, & Bosch, 2018). Issue tracking systems are useful ways of tracking issues when the number of issues becomes large and when a lot of people have to access and input data on them (Serrano & Ciordia, 2005).

By analysing historical data from an issue tracking system one can identify trends in the evolution of the debt and provide insights into if the amount of debt items increase beyond a threshold (Digkas, Lungu, Avgeriou, Chatzigeorgiou, & Ampatzoglou, 2018). These insights can thus support software practitioners in better understanding their own projects evolution. Then the question is how this information best can be presented.

Software visualization uses visual representations to present information that is often complicated to analyse (Z. Li et al., 2015). As visualization techniques have been shown to support the process of software understanding, Rios, de Mendonça Neto, and Spínola, 2018 and Brown et al., 2010 argues that visualization techniques should be further investigated in relation to technical debt.

The above findings reveals that the relationship between technical debt and security debt needs to be studied further in order to expand knowledge about the impact technical debt has on security. This will be done by analyzing the evolution of technical debt and security debt in issue trackers. There exists few studies on the evolution of technical debt in issue trackers (Tan, Feitosa, & Avgeriou, 2022b) which indicates the need for further investigations. The purpose of analysing the debt evolution in issue trackers is to understand and investigate the amount of issues that have been fixed and the time required to fix these issues. Moreover, as the prioritization activity is a crucial part of the debt management (for deciding which items to fix), I want to figure out how these insights can be visualized to support the planning and prioritization of technical debt and security debt. I, therefore, aim at answering the following three research questions:

RQ1 How different is the amount of fixed issues for technical debt and security debt?

RQ2 How different is the lead time for technical debt and security debt?

RQ3 How can visualizations of historical data in issue trackers support prioritizing the repayment of technical debt and security debt?

For the first two research questions I have analyzed a data set consisting of technical debt issues from a company. While the first research question focuses on the amount of fixed issues and the types (technical debt and security debt) in order to identity which issues are fixed more often, the second research question focuses on the survival of the various issues. The results from these two questions are used as a starting point when designing and developing the different visualizations. For the last research question, my contribution can be considered a first feedback from relevant stakeholders (e.g. managers, developers, architects) within one company. The visualizations are implemented as a dashboard in Jira. Then, the artefact is demonstrated and evaluated within two iterations.

1.1 Structure

- **Chapter 2 - Background**
  The background explains the concepts of technical debt and security debt along with how the debt is managed. Further, issue tracking systems and debt evolution analysis is described. Lastly, I look into software visualization and visualization techniques suggested in the context of technical debt.
- **Chapter 3 - Methodology**
  The methodology describes the Design Science Research process, the data collection methods and data analysis, as well as research ethics.

- **Chapter 4 - Results from quantitative data analysis**
  This chapter presents the results for the two first research questions.
- **Chapter 5 - Iterations**
  This chapter describes how the artefact has been developed and refined throughout two iterations. The findings from the evaluations are presented to answer the last research question.
- **Chapter 6 - Artefact**
  This chapter describes the technology used for the development of the artefact and the functionalities of the artefact.
- **Chapter 7 - Discussion**
  The findings from Chapter 4 and 5 are discussed together with related work. The discussion is structured according to the research questions. Finally, contributions, implications for research and practice are presented as well as validity and limitations.
- **Chapter 8 - Conclusion**
  In this last chapter, this study is concluded.

Chapter 2

Background

In this chapter I will present the background for the topics of this thesis. In the first section I describe the concept of technical debt. Then I present the concept of security debt as well as the framework used for classifying the security issues. Further, I look into debt management. Then I describe issue tracking systems and previous literature related to debt evolution analysis. Lastly, I look into software visualization and visualization techniques suggested in the context of technical debt.

2.1 Technical debt

Technical debt is a metaphor used for technical trade offs that can be beneficial in short term but may hurt the health of the software system in the long term (Z. Li et al., 2015). The metaphor was originally introduced by Ward Cunningham explaining to a non-technical stakeholder why the code refactoring took some time (Kruchten, Nord, & Ozkaya, 2012). From the original description: “not quite right code which we postpone making it right” (Kruchten et al., 2012; Sneed, 2014), the term has expanded with time.

Fowler, 2009 studied the metaphor of technical debt and divided it into four parts: reckless/prudent and deliberate/inadvertent debt.

- The debt is **prudent-deliberate** when the team knows that they are taking on a debt and thus puts some thought as to whether payoff for an earlier release is greater than the costs of paying it off.
- The debt is **reckless-inadvertent** when the team is ignorant of design practices and takes on the reckless debt without realizing how much hock it's getting into.
- The debt is **reckless-deliberate** when the team knows about good design practices but decides to go "quick and dirty" because they think they can't afford the time required to write clean code.
- The debt is **prudent-inadvertent** when the team understands what would have been the best design approach, then the team also realizes that they have an inadvertent debt.

In this thesis I'm looking at the technical debt that has already been identified. This means that according to Fowler's quadrant I'm looking at the deliberate technical debt and not technical debt that people do not know of.

Both in the research and industrial community, the metaphor of technical debt has been further developed to contain different types of technical debt as well as different ways to manage it (Rindell & Holvitie, 2019). Alves et al., 2016 and Rios et al., 2018 presents 15 types of technical debt where the most mentioned types are design, code, and architectural debt. The fact that technical debt types have expanded over time indicates that new fields are being included (Alves et al., 2016). One of the fields that have attracted more attention as technical debt have become more mature as a concept is security debt.

2.2 Security debt

Although there exists several studies investigating different software related factors for indicating security risks, little attention has been given on technical debt as an indicator (Siavvas, Tsoukalas, Jankovic, Kehagias, & Tzovaras, 2020). Based on the strong relationship found between technical debt and the security level of a software product, Siavvas et al., 2019 argues that technical debt is an factor that should be looked at in relation to indicating security risks.

The concept of security debt is derived from the technical debt metaphor (Martinez et al., 2021). Based on the assumption that security issues typically are linked to the internal quality, Rindell, Bernsmed, and Jaatun, 2019 explained security debt as technical debt that contains a security risk. In another study, security debt is divided in two parts: the term is first used for technical debt that has been identified through security verification or validation methods. Secondly, security debt is used for debt that has incurred through technical debt in a security critical software component (Rindell & Holvitie, 2019).

One major difference between technical debt and security debt is that security debt is highly related to security risks (Martinez et al., 2021). Security risk is defined by Firesmith, 2003 as: *"the potential risk of harm to an asset due to attacks"*. The risk is usually calculated by multiplying the negative impact of the harm with the likelihood of the harm occurring (Firesmith, 2003).

Recently, Kruke, 2022 presented a formal definition of security debt: *"a set of design or implementation solutions that hinder or has the potential to hinder the achievement of a system's optimal/desired/required security goal"* (p. 45). My thesis is using this definition and work as a basis for security debt. The reason for choosing this definition is that my research is done in the same company, which ensures that the definition holds in the same context.

2.2.1 Framework used to classify security debt issues

Firesmith, 2003 presents a large hierarchical taxonomy of quality factors and subfactors with the main function of providing a context for safety, security, and survivability. He defines a quality factor as *"a high-level characteristic or attribute of something that captures an aspect of its quality"* (p. 7).

In order to clearly distinguish between security and safety, security is defined using the term *malicious* whereas safety is defined using the term *accidental*:

- "Security is the degree to which **malicious** harm is prevented, reduced, and properly reacted to" (p. 11).

- "Safety is the degree to which **accidental** harm is prevented, detected, and properly reacted to" (p. 13).

Security is about attacks, while safety is about accidents. Having said that, attacks (security) may cause security risks that in turn may cause accidents. On the other hand, accidents (safety) can result in security vulnerabilities that can be exploited by attacks.

Security as a quality factor

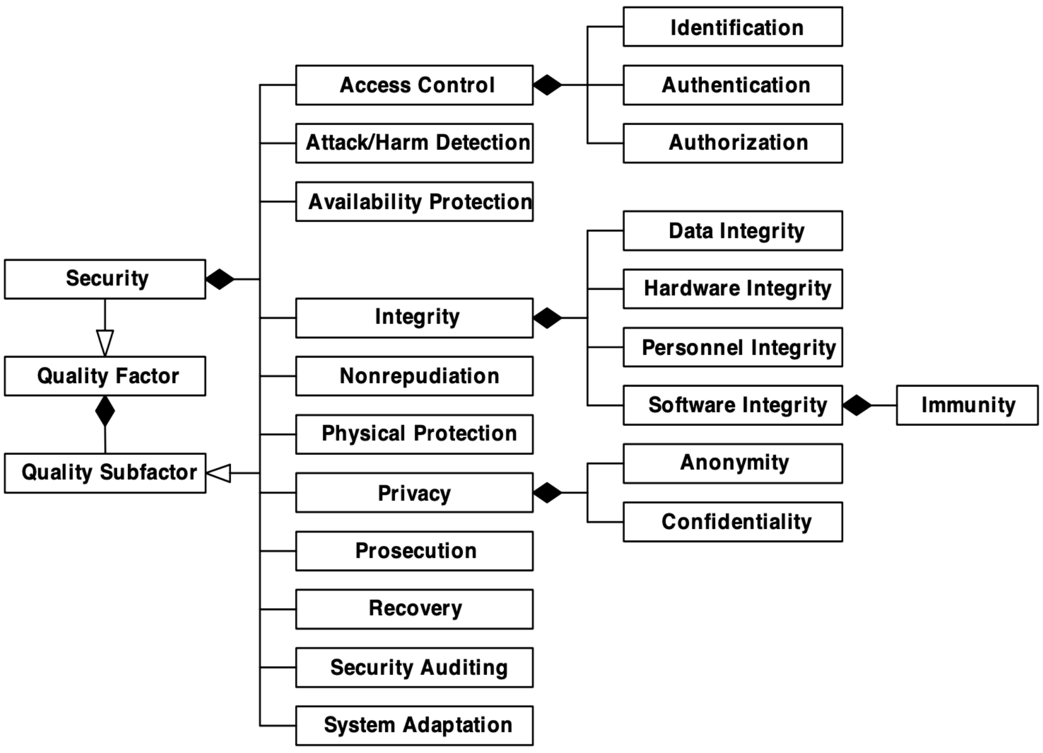


Figure 2.1: Model that hierarchically structures subfactors of security

Security is most commonly defined in terms of availability, integrity and privacy. But it is important to note that security is a relatively complex concept that cannot be adequately addressed simply using these terms. Therefore, Firesmith, 2003 expressed the need for a broadly accepted industry standard taxonomy of security and its quality subfactors.

Figure 2.1 shows how security is broken down into different quality subfactors in the taxonomy presented by Firesmith, 2003. This model has been the starting point for classifying the security debt issues in the data set I have looked at.

2.3 Debt management

As technical debt and security debt not can be totally eliminated in practice, it needs to be actively managed in order to be kept at an acceptable level (Z. Li et al., 2015). Different methods and tools have been used, proposed and developed for managing the debt.

2.3.1 Technical debt process

Technical debt management consists of a set of activities to either prevent potential technical debt from being incurred or to deal with existing technical debt to keep it under a reasonable level. Z. Li et al., 2015 presents eight technical debt management activities: identification, measurement, prioritization, prevention, monitoring, repayment, representation/documentation, and communication.

In this thesis I'm looking at the issues that are already identified and documented through an issue tracking system. Even though this thesis focus on the repayment action, how the repayment is done is out of scope. Because of this, I will not go into the identification, prevention and repayment activities.

Measurement

The measurement activity is related to quantifying the benefit and cost of known technical debt in a software system. The most studied method for measuring technical debt is to calculate technical debt through mathematical formulas or models (Z. Li et al., 2015). Additionally, Z. Li et al., 2015 mentions five other suggested approaches for measuring technical debt; code metrics, human estimations, cost categorization, operational metrics, and solution comparison. This activity is relevant for this thesis because by measuring the issues that are tracked one can make better decisions of which items to fix in the future (Guo et al., 2011).

Prioritization

The prioritization activity is done by ranking the identified debt according to which items should be repaid first and which items can be allowed until later releases. The studied approaches for prioritizing technical debt are cost/benefit analysis, high remediation cost first, portfolio approach, and high interest first (Z. Li et al., 2015). The prioritization activity is especially relevant for this thesis as the debt documented in issue trackers can provide insights into how technical debt and security debt has been prioritized in the past.

Implementing new features are often prioritized over refactoring technical debt. Moreover, technical debt that is not directly related to the implementation of new features are often postponed (Martini, Bosch, & Chaudron, 2015). Vathsavayi and Systä, 2016 emphasizes this: "Deciding on whether to spend resources for developing new features or fixing the debt is a challenging task".

Monitoring

The monitoring activity is done to see the changes of the cost and benefit of unresolved technical debt over time. Some of the suggested approaches for monitoring technical debt is to measure and track changes over time, and to define thresholds for related quality metrics and warnings if the thresholds are not met (Z. Li et al., 2015). By detecting trends in the evolution and give warnings when the items increase beyond a threshold, the monitoring activity helps to identify and prioritize repayment actions where technical debt items are fixed (Digkas et al., 2018). The monitoring activity is relevant for this thesis as the artefact created visualizes the evolution of technical debt and security debt over time.

Communication

The communication activity refers to making the identified technical debt visible to stakeholders so that it can be discussed and further managed. Different communication approaches suggested in the literature of technical debt is a backlog, different visualizations (such as code metric visualization and dependency visualization), and a dashboard displaying technical debt items, types and amount (Z. Li et al., 2015). This activity is relevant for my thesis because the visualizations that I create aims to improve the communication in the decision-making process related to the prioritization of technical debt and security debt.

Documentation

The documentation activity is a way of representing technical debt in a consistent manner to address the concerns of specific stakeholders (Z. Li et al., 2015). According to Z. Li et al., 2015, all of the included studies that suggests an approach for documenting technical debt is doing it by listing technical debt items along with different information (such as ID, issue type, author, etc.). The documentation activity is relevant for my thesis as the technical debt that I look into is already documented through an issue tracking system.

2.3.2 Security debt process

In order to make an informed decision on whether to take on the debt or not, the security risk must be identified and managed (Rindell & Holvitie, 2019). Kruke, 2022 argues that the eight activities found by Z. Li et al., 2015 applies for security debt management as well. In addition, she highlights three activities that were specifically mentioned for the security debt process: threat modeling, the use of a bug bounty program, and security testing performed by the company's security team. As these activities falls under the prevention and identification activities, they will not be further elaborated.

Measurement and monitoring

Security management activities includes the creation of appropriate security assurances to measure and monitor the accumulation of security risk. The technical debt management activities covers these areas of the security management

(Rindell & Holvitie, 2019).

Prioritizing

The accuracy of a risk assessment is crucial for the prioritization of security debt. This is because while the technical debt is mostly prioritized according to the principal and interest, the security items are assessed primarily by the risk (Rindell & Holvitie, 2019). Kruke, 2022 also finds in her study that most of the participants prioritize the security debt by calculating the risk severity.

Communication

The main reason for communicating security debt items is to make the risk visible and to create some discussion about it. The security issues are often communicated in the form of a security operations guide or an incident plan (Rindell & Holvitie, 2019). Kruke, 2022 points out that some of the participants highlighted the importance of communicating security debt, not only inside the team but also outside the team.

Documentation

When a security risk is known of, the debt is documented (Rindell & Holvitie, 2019). The documentation of security debt is not very different from the technical debt (Kruke, 2022). Kruke, 2022 found, in her study, that both technical debt and security debt were documented in the same backlog as everything else. The difference were that while technical debt is often tagged with "NFR" (non functional requirement), security debt is often labeled with a "security" tag in addition to the "NFR" tag.

2.4 Issue tracking systems

Incurring technical debt can be an investment if the debt is kept visible and under control (Z. Li et al., 2015). When working on a software project one have to manage the issues that are found. In order to manage the debt one must know of the debt, and in order to know of the debt one must track it. If the issues are not tracked they will remain invisible (Martini et al., 2018). Issue tracking systems are useful ways of tracking issues when the number of issues becomes large and when a lot of people have to access and input data on them (Serrano & Ciordia, 2005).

Issue tracking systems are repositories that are used to support the software development process (Ortu, Destefanis, Kassab, & Marchesi, 2015). By hosting development tasks (like bugs and new features), the system supports maintenance activity. For all of the stored items, the system provides information like a description, status of the issue, priority, as well as comments and attachments by developers to discuss the task at hand (Ortu, Destefanis, Murgia, et al., 2015).

According to Serrano and Ciordia, 2005, there exists more than 70 issues and bug tracking systems. Some popular issue tracking systems are Plutora, BugZilla, Backlog, and Jira (Mostowski, Kuder, Filipczak, & Rutkowski, 2018). In this thesis I'm looking at issues that are documented through the Jira issue tracking system as the company uses Jira to track their technical debt issues.

2.4.1 Jira

The Jira issue tracking system has become popular within the last years (Ortu, Destefanis, Murgia, et al., 2015), and is one of the most common issue tracking systems adopted by companies (Ortu et al., 2016). Originally, Jira is a bug and issue tracking system that were developed by Atlassian in 2002. Currently, it is used by 65 000 companies over the world (Atlassian, 2022c).

As an issue tracking system, Jira contains the standard issue tracking information mentioned above, as well as allowing for customized fields. In addition to tracking issues, Jira has grown into a powerful work management tool for all kinds of use cases (Atlassian, 2022c). Jira provides the possibility of tracking progress, managing the backlog, and to plan sprints (Ortu, Destefanis, Murgia, et al., 2015). Moreover, Jira offers unique features like a project management system and the Jira agile kanban board (Atlassian, 2022c).

2.5 Analyzing debt evolution

One major issue when dealing with software development is how to decide about future changes. "What evolution should the software system undergo, and in which sequence?" (Kruchten et al., 2012). This evolution is, according to Kruchten et al., 2012 constrained by cost in most cases, i.e. the resources available to apply to making these changes.

During software evolution, technical debt is being incurred and paid back, sometimes in the same day and sometimes ten years later (Digkas et al., 2018). One important aspect of technical debt is how the debt evolves over time and how it is paid back (Tan, Feitosa, Avgeriou, & Lungu, 2020). While it has been empirically proven that incurring technical debt negatively affects the project's quality, it is unwanted to totally eliminate the debt as the investment to reduce it would be extremely inefficient (Digkas, Lungu, Chatzigeorgiou, & Avgeriou, 2017).

There have been several studies in the literature investigating how technical debt in source code accumulates over time and the consequences of this accumulation for software maintenance (Digkas et al., 2018). This type of knowledge can be useful in terms of encouraging refactoring activities when the amount of technical debt increases fast or for preventing the accumulation of new technical debt if repayment strategies are too costly (Tan et al., 2020).

To the best of my knowledge, there exists only two studies that investigates the evolution of technical debt in issue trackers. Y. Li, Soliman, and Avgeriou, 2020 explores the identification and remediation of self-admitted technical debt in Jira. Tan et al., 2022b combines the sources of issue trackers and source code to study the life cycle of technical debt items. Aside from this, the studies looking into issue trackers in relation to technical debt is focusing on the identification of technical debt items.

2.5.1 Analysis of debt backlog

The technical debt backlog helps teams in remembering to take care of technical debt that would otherwise be invisible and overlooked (Martini et al., 2018). In this thesis I'm interested in how the debt items that are already documented in a backlog can be analyzed in order to give valuable information to relevant stakeholders.

For the first research question I focus on the amount of debt that is fixed and the types of issues that are fixed. For the second research question I focus on the survival of the various issues to find out after how much time the issues are fixed. Then the question is how these insights can be used to support the planning and prioritization of technical debt and security debt. Because of this, my last research question aims to find out how visualizations of historical data in issue trackers can be used for this purpose.

2.6 Software visualization

As vision is the most used sense by humans, software visualization techniques are increasingly researched. Software visualization techniques have been investigated in software engineering to help in understanding, maintaining, testing, and evolving software systems (Alves et al., 2016). These visualization techniques refers to the mapping of a software artefact (such as lines of code or method) to graphical representations (Mendes, Gomes, Gonçalves, & et al., 2019). As large amounts of data are produced during the evolution of a software, the use of visualizations can make the understanding of the system's evolution easier (Novais, Torres, Mendes, Mendonça, & Zazworka, 2013).

Based on the fact that software visualization has been shown to benefit the process of software understanding, Rios et al., 2018 and Brown et al., 2010 argues that the use of software visualizations in relation to technical debt needs further investigation.

2.6.1 Visualizing technical debt

Software visualization techniques can support the developer in the identification and/or management of different types of technical debt in software projects (Alves et al., 2016). Mendes et al., 2019 claims that software visualization techniques are a promising way to deal with technical debt items.

There exists few studies that has proposed visualization techniques related to the identification of technical debt. The suggested techniques to support the identification are: flags in code, 2D maps, scatterplot and correlation matrix, time range, timeline and treemap (Alves et al., 2016). In the context of technical debt management, software visualization techniques are more studied and the most proposed techniques are dependency matrix, bar graph, and pie chart format (Alves et al., 2016). Besides this, Alves et al., 2016 and Novais et al., 2013 points out that the different types of visualization techniques already suggested in other contexts in software maintenance and evolution should be investigated for the management of technical debt.

In the context of agile development, burn down and burn up charts are usually used to keep track of progress. Burn up charts shows the functionality implemented over a time period, while burn down charts shows the work remaining to reach a specific goal over a period of time. Other visualizations used for progress tracking are concerned with the velocity of the team and estimating efforts (Paredes, Anslow, & Maurer, 2014). Additionally, treemaps, timelines, and pie charts are also commonly used within agile development (Paredes et al., 2014).

In addition to visualization techniques, another useful approach when dealing with large amounts of complex data are software metrics. Lanza, 2001 combines these two approaches, i.e., software metrics are visualized allowing for quickly understanding the evolution. Moreover, Pinzger, Gall, Fischer, and Lanza, 2005 points out that software metrics should be communicated to highlight trends in the large amounts of data, because by spotting these trends one can get a thorough understanding of the current state and thus further focus on maintaining the relevant entities.

2.6.2 Visualizations in dashboards

In dashboard, the information is organized and presented in an easy to read manner. The use of dashboards is beneficial in terms of letting managers and developers access a summarized version of the project's data without having to go through log files to gather and understand the information. By using visualization techniques on dashboards one can improve the awareness of their own project and thus make them act upon the conflicts that arise (Paredes et al., 2014).

In this thesis I aim to create a dashboard that makes the planning and prioritization of technical debt and security debt easier. Some of the mentioned visualization techniques are used as an inspiration for designing the visualizations. Additionally, different software metrics (such as number of open issues and lead time) are included.

Chapter 3

Methodology

In this chapter I will present the methodology used in the thesis. First, I will present the research design and how this design was applied to my research. Then I will describe the different methods of data collection. Further, I will explain how the data was analyzed. Lastly, I will present research ethics.

3.1 Company context

The company used in this study is a multinational company with a growing number of independent subsidiaries and is currently found in over 20 countries. Each of the independent subsidiaries use global processes such as the annual assessment of the technical debt process. The company use an index application that contains multiple areas where they measure the maturity of their products. Two of these areas are Architecture and Technology, and Security. Under these areas each product that the company owns has assigned one target level. Each level has assigned the interval of penalty points that you can have for that level. These penalty points you get if you do not perform several actions, and each action has a different rating depending on the importance of it.

3.2 Design Science Research

A central part of my research is to find out how visualizations can support the planning and prioritization of technical debt and security debt. As this can be achieved by creating and evaluating an artefact, the research design chosen for this study is Design Science Research.

Design Science Research can be considered a problem solving paradigm (Brocke, Hevner, & Maedche, 2020; Hevner et al., 2004). Hevner et al., 2004 argues that Design Science is about creating and evaluating innovative artefacts to solve specific problems. An artefact is defined by Johannesson, 2014 as a humanly-made object used to address a practical problem. A common distinction within Design Science Research is between four types of artefacts: constructs, models, methods, and instantiations (Johannesson, 2014).

Instantiations are working systems that can be used in a practice (Johannesson, 2014). The artefact that I aim to create will work as a dashboard implemented in the existing Jira platform. Because of this, the artefact provided by my study can be considered an instantiation.

An important step following the Design Science Research methodology is to evaluate the created artefact. This is done to determine if the artefact is solving the explicated problem. Different strategies have been suggested to evaluate an artefact but one common distinction is between ex ante and ex post evaluations. Ex ante evaluation is evaluation that is conducted before the artefact is fully developed, while ex post evaluation requires that the artefact is working (Johannesson, 2014).

Hevner et al., 2004 created seven guidelines to support researchers in developing, improving, and evaluating artefacts. In Chapter 7 I will go through how these guidelines have been followed throughout my project.

3.2.1 Design Science Research process

I have applied Peffers, Tuunanen, Rothenberger, and Chatterjee, 2007 Design Science Research Methodology in my research. The procedure follows 6 steps: problem identification and motivation, definition of the objectives for a solution, design and development, demonstration, evaluation, and communication. A visual representation of the research design is presented in Figure 3.1 and explained below.

Figure 3.1 shows an arrow above the six steps. Throughout my project I've had weekly meetings with my supervisors where we have discussed methodology, results and interactions with the company. Company stakeholders have been involved in most of these meetings which has given me the opportunity of getting continuous input and feedback from relevant stakeholders within the company.

Step 1 is the activity of defining the specific problem and justifying the value of a solution (Peffers et al., 2007). In this first step I got input from the company stakeholders (involved in our meetings) to define the research topic for this thesis. Additionally, I reviewed the current literature on technical debt and security debt. This revealed that even though visualization techniques are shown to benefit software understanding, there exists few studies related to visualizations of technical debt. Based on the findings from the literature review it is reasonable to assume that the planning and prioritization of technical debt and security debt is considered a challenge. These findings together with feedback from the company stakeholders were the base from where I defined my research questions. This will further be used to develop the artefact in order to provide a solution.

Step 2 is the activity of creating the objectives of a solution based on the problem definition (Peffers et al., 2007). In this step I conducted a document analysis where I looked into a data set containing technical debt issues from the company's Jira backlog. The document analysis helped me in finding out what data was available in the company. Furthermore, I performed some statistical tests on the data. These tests serve as a starting point for the rest of the thesis as I got to explore the problem area and understand how the teams prioritize and fix technical debt and security debt within the company. A more detailed explanation of the document analysis and the statistical tests will be presented later in this

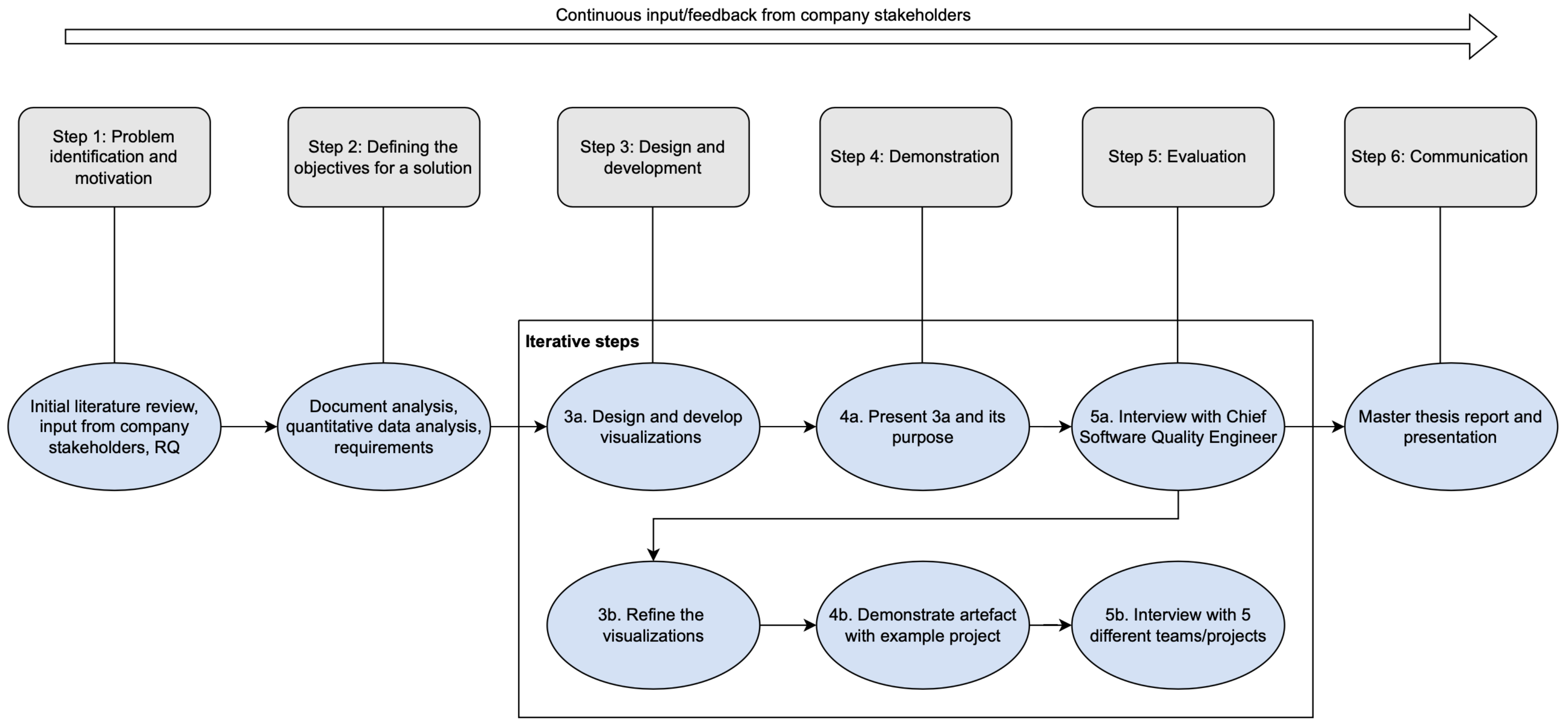

Continuous input/feedback from company stakeholders
Step 1: Problem identification and motivation
Step 2: Defining the objectives for a solution
Step 3: Design and development
Step 4: Demonstration
Step 5: Evaluation
Step 6: Communication
Initial literature review, input from company stakeholders, RQ
Document analysis, quantitative data analysis, requirements
Iterative steps
3a. Design and develop visualizations
4a. Present 3a and its purpose
5a. Interview with Chief Software Quality Engineer
3b. Refine the visualizations
4b. Demonstrate artefact with example project
5b. Interview with 5 different teams/projects
Master thesis report and presentation

Figure 3.1: Design Science Research process

chapter. With the findings from the statistical tests in mind, I discussed different ideas of how to solve the explicated problem together with the company stakeholders involved in our meetings. This resulted in a set of functional requirements (see section 5.1) which was the base from where the visualizations were designed and developed.

The three next steps: design and development, demonstration, and evaluation are done in an iterative manner with two iterations as shown in Figure 3.1. After completing the evaluation in the first iteration I went back to redesign the artefact as a result of the feedback I got. A more detailed explanation of these steps will be described in chapter 5.

Step 3 is the activity of creating the artefact (Peffers et al., 2007). In the first iteration I designed and developed the different visualizations based on the requirements proposed by the company stakeholders. In the second iteration I improved the visualizations based on the feedback provided in the first evaluation.

Step 4 is the activity of demonstrating the artefact in order to solve one or more instances of the explicated problem (Peffers et al., 2007). In the first iteration I conducted an interview with a Chief Software Quality Engineer. In the interview I presented the findings from the statistical tests as well as the different visualizations and their purposes. In the second iteration I demonstrated the artefact and interviewed different roles (e.g. managers, developers, and architects) from five teams/projects within the company.

Step 5 is the activity of evaluating the artefact to find out how well it supports a solution to the explicated problem (Peffers et al., 2007). The goal for the first evaluation were to find out about the usefulness of the visualizations and what could be improved. This evaluation led to refining the visualizations. The evaluation step in the second iteration were done to find out to what extent the evaluation criteria **understandability** and **usefulness** were met.

Step 6 is the activity of communicating the problem and its importance (Peffers et al., 2007). This last step is this master thesis report.

3.3 Methods for data collection

In this section I will describe the different data collections methods used in the research. Although it is common to use a single data collection method in a research project, it can be helpful to use more than one method to get a broader view. The approach of combining research strategies and methods is called the mixed method approach (Johannesson & Perjons, 2014). I have used the mixed-method approach in my project by using different sources of data.

The purpose of choosing this mixed-method approach was to first explore how the different teams works with technical debt and security debt using document analysis and quantitative data analysis. Then, based on the findings from the quantitative data analysis together with input from the company stakeholders, I developed an artefact and conducted interviews with different relevant stakeholders. The reason for using both quantitative and qualitative data was that the visualizations could best be developed after a preliminary investigation on how the teams manage technical debt and security debt.

3.3.1 Document analysis

Document analysis is a systematic procedure for reviewing or evaluating documents. By reviewing documents one can gain insights into the context where research participants operate. Documents can provide information as well as historical insights. Such information can help researchers understand past events (Bowen, 2009).

Using documents as a data source means that a great deal of data can be collected in a shorter period of time and more inexpensively than what would be the case with questionnaires or interviews (Brocke et al., 2020).

The document I have gained access to is a data set consisting of technical debt issues from the company's Jira backlog. One of the company stakeholders extracted the issues and imported it to my project in TSD [1] (Tjeneste for Sensitive Data),

[1] https://www.uio.no/tjenester/it/forskning/sensitiv/

owned by the University of Oslo, operated and developed by the TSD service group at the University of Oslo, IT-Department (USIT).

Data set description

- The data set consists of 10970 technical debt issues
- The issues are created from November 2012 until April 2022
- The data set reports information about the *creation date*, *resolution date* (if resolved), *summary*, *priority*, *status*, *resolution*, and *labels*
- There are four different priority levels: *low*, *medium*, *high*, and *critical*
- The issue can be in one of these four statuses: *open*, *in progress*, *in review*, and *closed*
- The different resolutions reported are: *done*, *fixed*, *won't do*, *won't fix* and *unresolved*

Excluded issues

I have removed issues lacking the required information, i.e. issues without a priority, issues that have been moved, duplicates, etc. Additionally I have excluded the issues that are created in April 2022 because there is a much higher amount of unresolved issues compared to the other months (16% vs. 50%). After removing these issues the data set consists of 10192 issues.

Table 3.1 shows the distribution of issues for the different priorities after the removal of the mentioned issues.

Resolution	**Low**	**Medium**	**High**	**Critical**	**Total**
Fixed	297	4906	619	135	**5957**
Not fixed	338	2864	182	24	**3407**
Won't fix	91	687	47	3	**828**

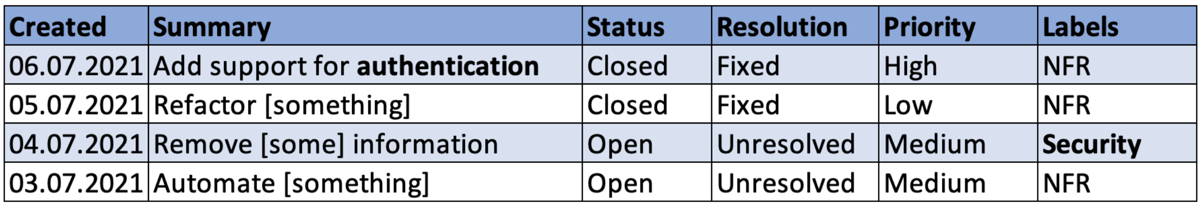
Table 3.1: Distribution of issues for the different priorities

Issue resolution

I distinguish between issues that are fixed and issues that are not fixed. The issues that are considered fixed are the issues that are closed with the resolution *"done"* or *"fixed"*. The issues that are closed without a resolution (*"won't do"* or *"won't fix"*) are not included in the fixed issues. The issues that are considered not fixed are the issues that are still open with the resolution *"unresolved"*.

Issue types

I have classified some of the issues in the data set as security debt. This has been done by searching in each of the issues summary and label fields. In these fields I have searched for the words proposed in the security model by Firesmith, 2003 (see Figure 2.1). For example, if the issue is labeled *"security"* or if the summary mentions *"authentication"* I have classified that item as security debt. The figure below shows four example items and how the classification is done.



Created	Summary	Status	Resolution	Priority	Labels
06.07.2021	Add support for **authentication**	Closed	Fixed	High	NFR
05.07.2021	Refactor [something]	Closed	Fixed	Low	NFR
04.07.2021	Remove [some] information	Open	Unresolved	Medium	**Security**
03.07.2021	Automate [something]	Open	Unresolved	Medium	NFR

Figure 3.2: An example showing how the security debt items are classified by searching in the summary and labels field. The colored rows shows the issues that are considered security debt. I have removed some information to keep the data from the company anonymous.

Different projects

The data set consists of data from 181 different projects. Some of the projects have very few issues, while others have a large amount of issues. 55 of these projects have both technical debt and security debt issues. Since a part of this thesis is to investigate the relation between technical debt and security debt, I have performed the statistical tests for both of these cases: 1) using all technical debt and security debt issues (without considering projects), and 2) using data only from the projects with both technical debt and security debt. This can be seen when reporting the results in Chapter 4.

3.3.2 Interview

Interviews are useful for collecting data from people with deep and unique information and knowledge about some domain. A big advantage of conducting interviews is that the researcher is allowed to go into depth in order to collect detailed and complex information (Johannesson & Perjons, 2014).

Interviews can be divided into three types based on the structure of the interview; structured, semi-structured and unstructured interviews. The semi-structured interview were chosen for all of the interviews as this opens up for partcipants to freely express their ideas (Johannesson & Perjons, 2014).

I have conducted interviews within both iterations. In the first iteration I interviewed a Chief Software Quality Engineer in the company in order to get some initial thoughts and improvements on the visualizations. In the second iteration, I refined the visualizations by including the improvements suggested in the first evaluation. Then, I interviewed different stakeholders from five teams/projects within the company. Because this can be seen as a first feedback regarding visualizing technical debt and security debt for the purpose of supporting the planning and prioritization I wanted to get feedback from different perspectives within a team/project.

To prepare for the interviews I made an interview guide. Because I conducted interviews in two iterations and these iterations had different goals I made two different interview guides (one for each iteration). In the first evaluation, the main focus were on the relevance of the visualizations and how these best could be shown, while the interviews in the second iteration focused on evaluating the understandability and usefulness of the refined visualizations. The interview guide used in the first iteration is included in Appendix A (8) and the interview guide used in the second iteration is included in Appendix B (8).

Before interviewing the participants in the second iteration I got some help formulating the questions to ask. I had a meeting with a Senior User Experience

Designer from the company where I showed my visualizations and told what I wanted to get out of asking the participants. This preparation supported me in the creation of the interview guide for the second iteration. One of the main things that were discussed in the meeting was how to evaluate the understandability of the visualizations. It was proposed to ask open questions about what the participant could see in the visualizations to get an impression of how it was perceived. Another thing that was pointed out in the meeting was the importance of getting candid and honest answers from the participants. This was especially important as I had developed the visualizations myself and then the participants could be cautious to avoid hurt my feelings. Because of this, I told the participants that they should say if they don't like or understand the visualizations because only then I can do something about it.

First evaluation

The participant in the first evaluation was selected in agreement with my supervisors. The participant is a Chief Software Quality Engineer and is therefore someone who knows a lot about how the prioritization of technical debt and security debt is done within the company.

This interview was conducted in the first iteration. The goal of this first evaluation was to find out about the usefulness of the visualizations and how they could be improved. Because the statistical tests were the starting point for the creation of the visualizations I explained to the participant which tests were performed and the results from the tests. The visualizations were demonstrated in a presentation showing one visualization on each slide.

This first evaluation can be considered an ex ante evaluation because the artefact was not demonstrated as an integrated solution in Jira. This was chosen as this type of evaluation is ideal for evaluations where an initial design or prototype is to be assessed quickly and inexpensive in order to obtain feedback for further improvement (Johannesson & Perjons, 2014).

Interviewing stakeholders within 5 projects/teams

In the second iteration I conducted interviews with different roles from five teams/ projects within the company as shown in Table 3.2. The participant in the first evaluation suggested different people that I could ask in this second iteration.

Interview	Referred to as	Role	# open TD
1	A1	Service architect	5
2	B1	Development manager	17
3	C1, C2	Backend developers	92
4	D1	Project manager	26
5	E1, E2, E3, E4	Software architects	883

Table 3.2: Table showing the participants in the second iteration. Interview 3 and 5 were conducted as focus group interviews. The number of participants in each interview is reflected in the "referred to as" column. The "referred to as" column shows how I will refer to the participants when including citations of what they have said. The four first interviews were from different teams and the last interview was four people from one project (meaning different teams within the same project) which is why the number of open issues are much higher.

Two of the interviews were conducted as a focus group interview with two and four participants. This was done because the participants in the same interview were from the same team/project within the company. By conducting a focus group interview, the participants can be more creative and more in depth as the participants can inspire each other to come up with ideas (Johannesson & Perjons, 2014). A disadvantage with this type of interview is that the participants can influence each other and thereby drive the discussion in a certain direction. Thus, this type of interviews might not be useful for understanding the views of every participant. As the participants in the same focus group interview has the same role and daily tasks within the same company, I think this disadvantage is greatly mitigated.

I wanted to demonstrate the artefact with the participant's own data, but as they then could be more concerned with discussing the actual numbers and not the visual representations, I decided, together with the company stakeholders, to use data from another project when demonstrating it. When firstly creating the visualizations I used the five biggest projects in terms of amount of issues for both technical debt and security debt. When demonstrating it to the different stakeholders I chose to use the middle project out of these five largest.

The evaluation criteria for these interviews were **understandability** and **usefulness** of the visualizations. In order to get data on the understandability I asked open questions about the visualizations before explaining them and their purpose. This was done to get an impression of how they perceive the different visualizations. Related to the understandability, I for example asked "What do you see here?" or "What do you think this visualization is showing?". To get data related to the usefulness of the visualizations I asked questions related to how it could help the participants in their work with technical debt and security debt, who would use it and when it would be used. Further, I asked the participants what is missing in the different visualizations to find out how they could be further improved.

The evaluations conducted in this second iteration can be considered an ex post evaluation as the artefact is fully working and implemented as a dashboard in Jira. Because the completed artefact is evaluated, this kind of evaluation is less likely to produce false positives (Johannesson & Perjons, 2014). Although the artefact is fully working, I did ask for and will report some suggested improvements for the artefact.

3.4 Data analysis

There are two main kinds of data analysis: quantitative and qualitative analysis. Quantitative data analysis works on quantitative data (numbers), while qualitative data analysis works on qualitative data (words, images, etc.) (Johannesson, 2014). In this section I will describe how the data has been analysed. Since I have both quantitative and qualitative data I will first explain the quantitative data analysis and then describe the qualitative data analysis.

3.4.1 Quantitative data analysis

When doing a quantitative data analysis one typically distinguish between descriptive statistic and inferential statistics. The difference between these two is that descriptive statistics intends to describe a given sample of data (for example mean, median, mode, etc.) while inferential statistics is used to draw conclusions about a population that the sample represents (Johannesson, 2014).

Johannesson and Perjons, 2014 presents four kinds of quantitative data that differ in terms of how they can be manipulated mathematically. Below is a short explanation of these types:

- **Nominal/categorical data** does not have a numerical value on which arithmetical operations can be performed.
- **Ordinal/ranked data** also denotes categories, but these categories can be ranked and there is an order between them.
- **Interval data** are ordered and the distance between two adjacent points on the scale is always the same.
- **Ratio data** are like interval data and additionally there is a zero to the scale.

In my research I have used inferential statistics in order to investigate the relationship between technical debt and security debt. This has been done using two-population hypothesis testing. For performing the statistical tests I have used the program Xlstat [2].

The data set that I have looked at consisted of technical debt issues from different projects within the company. All projects with security debt did also have technical debt. On the other hand, projects with technical debt might not have security issues. Because not all of the projects had security issues, I performed the statistical tests for the following two cases:

1. All technical debt and security debt without considering projects
2. Technical debt and security debt only from projects having both security debt and technical debt

[2]https://www.xlstat.com/en/

The first case was to consider all of the issues in the data set to get a complete overview and the second case was to only compare the projects that has both security debt and technical debt as mentioned in the research questions.

Chi Square test of independence

I wanted to explore the relationship between issue type (security debt and technical debt) and whether or not the issue is fixed in order to see if one of the issue types are fixed in higher quantities than the other and thus are considered more important to the team/project. For this purpose, I have performed the Chi-Square Test of Independence. The data used for this test is nominal/categorical data (issue type and resolution) and the samples are independent.

The reason for choosing this test is that the Chi-Square Test of Independence can be used to study relationships in categorical data. If the variables are found to be unrelated, they're declared independent. If they're found to be related, they're declared dependent (Rumsey, 2009).

The Chi-Square Test of Independence is first calculating what is expected to be seen in each of the cells in the table if the variables are independent. Then the test compares these expected cell counts to what you observe in the data and compares them using a Chi-Square statistic (Rumsey, 2009).

Table 8.1 (Appendix C) shows the counts for each issue-fixing combination. All technical debt issues (without considering projects) are denoted with *TD (all)*. The technical debt issues that are only from the projects that has both technical debt and security debt are denoted with *TD (projects)*. Lastly, the security debt issues are denoted with *SD*. The expected counts represents the number of fixed or not fixed issues if there are no relationship.

I have performed the Chi-Square Test of Independence for all of the priorities together (overall) as well as for each priority (low, medium, high and critical). For the critical security issues, the expected count for *not fixed* issues are 3 (see Table 8.1). Because a requirement for this test is that there are more than five expected counts for each combination of the two variables, the result for the issues with critical priority is not as reliable as the other results.

I have used a 5% significance level, i.e., the null-hypothesis was rejected if the p value was smaller than 0.05. The following is my null hypothesis and alternative hypothesis:

- H0: There is no relation between issue type and fixing
- HA: There is a relation between issue type and fixing

Mann-Whitney U test

I wanted to find out if the lead time for fixing an issue were different for technical debt and security debt. For this purpose I have performed the Mann-Whitney U test. The data used for this test is ordinal data (lead time). I have one independent variable (issue type) and one dependent variable (lead time).

The reason for choosing this test is that the Mann-Whitney U test can be used to test the null hypothesis that two samples come from the same population or similar populations (Beatty, 2018). I could have used an independent t-test for this purpose, but since the data that I have is not normally distributed (Shapiro-Wilk test was significant for all cases) I have used the non-parametric test Mann-Whitney U test. This test doesn't require the data to have a specific distribution (Rumsey, 2009).

I removed outliers in the data set using the interquartile range technique. This was done because outliers are patterns which are not in the range of normal behaviour (Vinutha, Poornima, & Sagar, 2018). The interquartile range technique works by dividing the data into three quartiles: Q1, Q2, and Q3. Then the interquartile range is calculated by Q1 and Q3. In the end, the upper and lower boundaries are found by using the interquartile range and then removed.

I have performed the Mann-Whitney U test for all of the priorities together (overall) as well as for each priority (low, medium, high and critical). I have used a 5% significance level, i.e., the null-hypothesis was rejected if the p value was smaller than 0.05. The following is my null hypothesis and alternative hypothesis:

- H0: The lead time for technical debt and security debt is equal

- HA: The lead time for technical debt and security debt is not equal

With lead time I mean the time required to fix the issue. The lead time is found by calculating the days between the *created* and *resolved* time stamps. When performing the test I did only used the issues that are actually fixed, i.e., the issue that are closed with a resolution ("done" or "fixed"). I have not included the issues that are closed without a resolution ("won't do" or "won't fix").

I have used Cohen's classification of effect size to determine the strength between the variables. The effect size is found by: $r = \frac{Z}{\sqrt{N}}$.

r > 0.3	small effect
r < 0.3 & r > 0.5	medium effect
r < 0.5	large effect

Table 3.3: Cohen's classification of effect size

3.4.2 Qualitative data analysis

All of the interviews were conducted and recorded in Zoom. Then, the recordings were transcribed and anonymized. I used thematic analysis to categorize the data in the transcribings.

Thematic analysis

Thematic analysis is used to identify, analyse, and report themes within data. A theme is a collection of something important about the data in relation to the research question. Additionally, the theme presents some level of patterned meaning within the data set (Braun & Clarke, 2006).

I analysed the interviews using themes. There are two common ways to identify themes in thematic analysis: inductive way or deductive way. For the inductive approach, the coding frames emerges from the data, while for the deductive approach the coding frames are pre-defined (Braun & Clarke, 2006).

Deductive approach

For both of the iterations, I used the deductive approach when coding the data. I chose to do it this way because I had already defined the topics I wanted to find out about in the two interview guides. Additionally, since I'm coding for specific evaluation criteria, this seemed to be the best choice. The themes were selected by the topics in the interview guides. Below is a figure (see Figure 3.3) showing an example of how the thematic analysis is done from the themes to the quotes from the interviews.

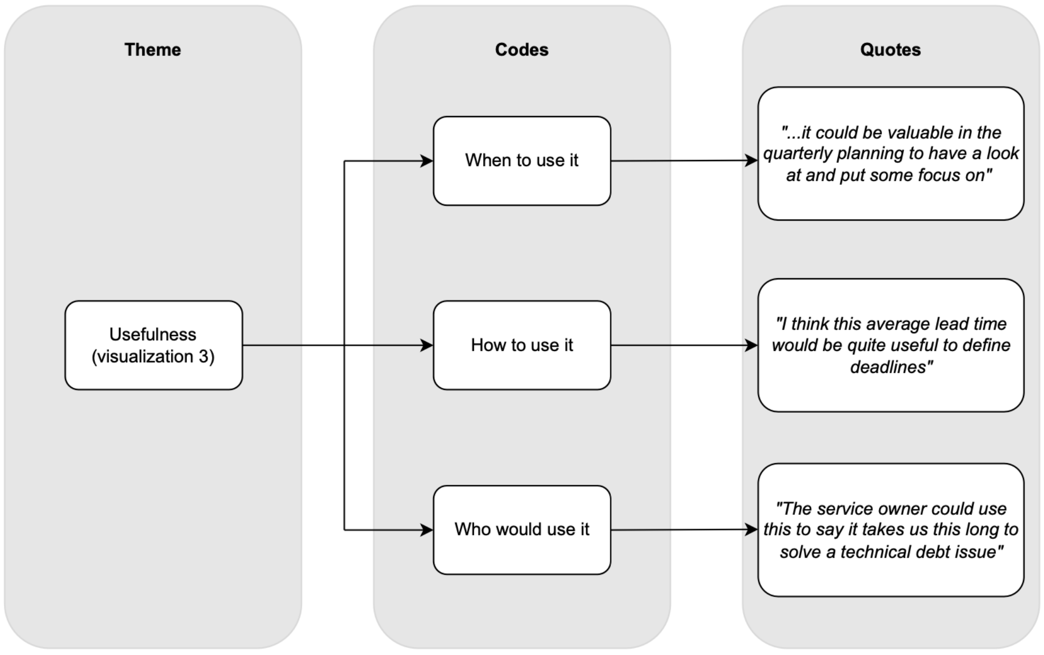

Theme
Codes
Quotes
Usefulness (visualization 3)
When to use it
How to use it
Who would use it
"...it could be valuable in the quarterly planning to have a look at and put some focus on"
"I think this average lead time would be quite useful to define deadlines"
"The service owner could use this to say it takes us this long to solve a technical debt issue"

Figure 3.3: An example showing the thematic analysis from theme to interview quotes

The themes I have used for the thematic analysis are shown under the evaluation section for each of the iterations in chapter 5.

3.5 Research ethics

The data set that I gained access to from the company was anonymized in advance by one of the company stakeholders. Even though there were no identifiable information, I considered the data to be in confidence data because it contained information about the company's issues. In accordance to the University of Oslo's data storage guide I created a project in TSD so that the data could be stored securely. Additionally I created a data management plan that was sent to NSD [3] (Norwegian Center for Research data).

The interviews were recorded. Before conducting the interviews I sent a notification form to NSD. In the notification form I uploaded the interview guides, a consent form as well as described how the recordings would be stored. Before starting the interviews I asked the participants if they were fine with me recording the meeting. I read the most important parts of the consent form before I asked them for a consent. After the interview I sent the full consent form to the participants by mail.

[3]https://www.nsd.no/en

Chapter 4

Results from quantitative data analysis

As described earlier, I have performed two different statistical tests to investigate the relationship between technical debt and security debt. The results from these tests will be presented in this chapter. The sections corresponds to the two first research questions described in Chapter 1.

As I have performed the test for the two cases described in the Methodology (section 3.4.1), the results are presented in this order for both presented tests:

1. All technical debt (*All TD*) and security debt (*SD*) without considering projects.
2. The technical debt (*TD from projects*) and security debt (*SD*) from only the projects having both technical debt and security debt

4.1 RQ1. How different is the amount of fixed issues for technical debt and security debt?

The aim of this research question is to quantify the amount of technical debt and security debt that is paid off and thus find out which issues are fixed more often. I considered 9357 Jira issues. Table 4.1 shows the distribution of issues for the different priorities. I tested the relation between issue type and resolution by performing the Chi-Square Test of Independence.

Issue type	Resolution	Low	Medium	High	Critical	Total
All TD	Fixed	238	4255	396	118	**5007**
	Not fixed	290	2655	157	23	**3152**
TD from projects	Fixed	223	3899	363	88	**4573**
	Not fixed	269	2335	151	12	**2767**
SD	Fixed	59	630	223	17	**929**
	Not fixed	48	195	25	1	**269**

Table 4.1: Distribution of considered issues by priority and resolution

Overall
As the obtained p value is $1.8E^{-27}$/$1.6E^{-24}$ we should reject the null hypothesis and accept the alternative hypothesis. In general, the Chi-Square Test of Independence shows that there is a relation between issue type and fixing.

Low
As the obtained p value is .057/.065, the null hypothesis cannot be rejected. For the issues with low priority, the Chi-Square Test of Independence shows that there is no significant relation between issue type and fixing.

Medium
As the obtained p value is $8.7E^{-17}$/$7.3E^{-15}$ we should reject the null hypothesis and accept the alternative hypothesis. For the issues with medium priority, the Chi-Square Test of Independence showed that there was a significant relation between issue type and fixing.

High
As the obtained p value is $1.08E^{-8}$/$3.19E^{-9}$ we should reject the null hypothesis and accept the alternative hypothesis. For the issues with high priority, the Chi-Square Test of Independence showed that there was a significant relation between issue type and fixing.

Critical
As the obtained p value is .23/.42, the null hypothesis cannot be rejected. For the critical issues, the Chi-Square Test of Independence showed that there was no significant relation between issue type and fixing, X^2=1.4/0.65, p=.23/.42.

The results from the Chi-Square Test of Independence are summarized in Table 4.2. This table shows that the results are not very different for the tests where all technical debt are included and where only the technical debt from the projects with security debt are included. At least, the difference does not impact if the result is significant or not.

To further look at the significant relations I have used a bar chart to graph each cell's contribution to the Chi-Square statistic (see Figure 4.1). This graph emphasizes the small difference for the considered technical debt issues.

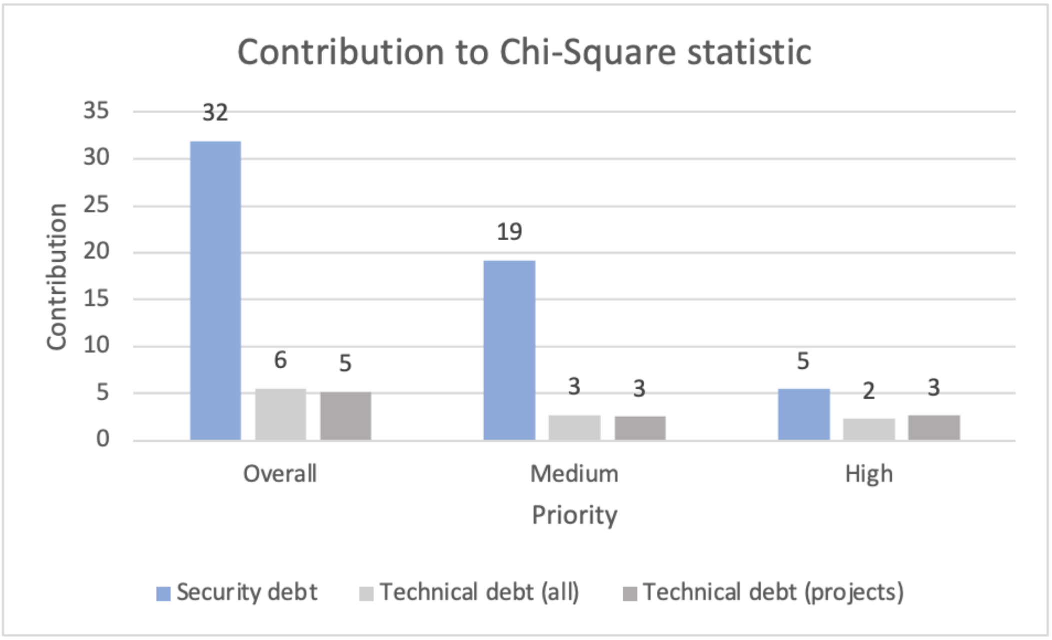


Figure 4.1: Contribution to the Chi-Square statistic. The contribution is rounded to its nearest integer.

The contribution graph shows that it is the security debt that contributes the most to the Chi-Square statistic and thus produce the statistical significance. The technical debt issues adds almost nothing. This means that, for these priorities, the security debt issues are fixed more frequently than expected. This can also be seen by looking at the statistical output for the observed and expected counts in the Table 8.1 (Appendix C). Moreover, I will show the percentage of fixes for all of the priorities in a bar chart.

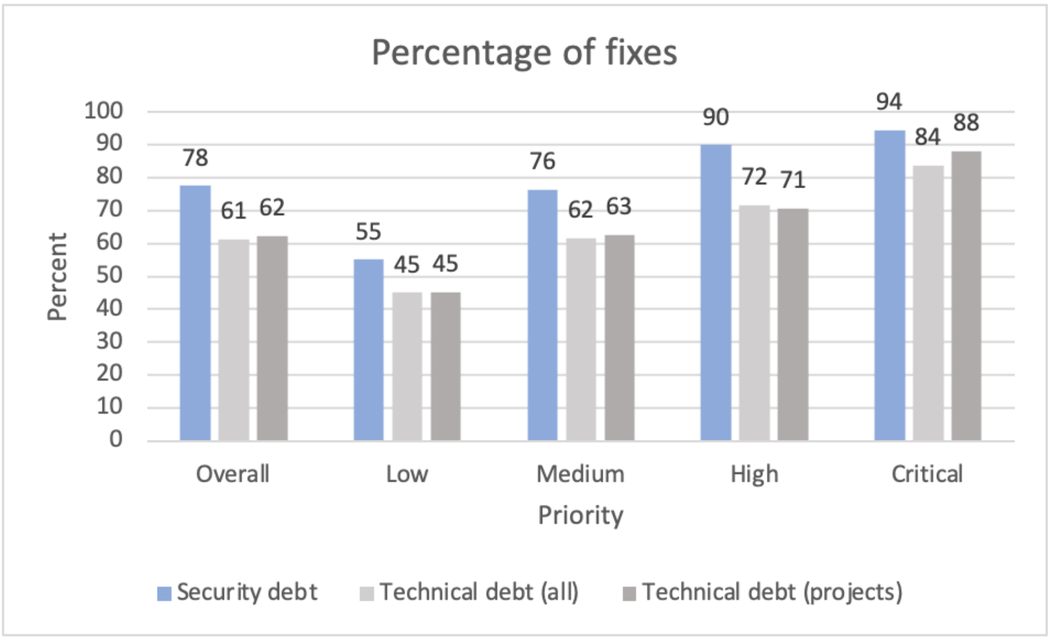


Figure 4.2: Percentage of fixes for each priority type. The number is rounded to its nearest integer.

Figure 4.2 shows the percentage of fixes for technical debt and security debt. By looking at this graph we can see can see the same observations as above: the percentage of fixes is higher for the security debt issues. This is also the case for the issues with low and critical priority even if the relation between issue type and resolution were not significant. In addition, this graph shows that the percentage of fixed issues goes up as the priority gets higher for both technical debt and security debt.

Priority	Included issues	Significant difference	p value	Interpretation
Overall	TD (all) and SD	Yes	$1.8E^{-27}$	**Security debt** is significantly **fixed more often** than **technical debt**
	TD (projects) and SD		$1.6E^{-24}$	
Low	TD (all) and SD	No	.057	There is **no relation** between issue type and fixing
	TD (projects) and SD		.065	
Medium	TD (all) and SD	Yes	$8.7E^{-17}$	**Security debt** is significantly **fixed more often** than **technical debt**
	TD (projects) and SD		$7.3E^{-15}$	
High	TD (all) and SD	Yes	$1.08E^{-8}$	**Security debt** is significantly **fixed more often** than **technical debt**
	TD (projects) and SD		$3.19E^{-9}$	
Critical	TD (all) and SD	Yes	.23	There is **no relation** between issue type and fixing
	TD (projects) TD (projects) and SD		.42	

Table 4.2: Results from the Chi-Square Test of Independence

4.2 RQ2. How different is the lead time for fixing technical debt and security debt?

The aim of this research question is to find out after how long time the issues are repaid and if there is a difference in the lead time for technical debt and security debt. For this research question I only considered the issues that are fixed which are 5955 Jira issues. I considered as outliers the data outside the interquartile range. After this, the data set contained 5470 issues. The distribution of the lead time is shown in Figure 4.3. I tested the differences of the distributions of lead time using the Mann-Whitney U Test.

Overall
As the obtained p value is 0.002/3.00E^{-8}, we can reject the null-hypothesis and conclude that there is a difference in the lead time for technical debt and security debt. In general, the Mann-Whitney U test indicates that the lead time is higher for security debt issues (Median=82) than technical debt issues (Median=77/63). The effect size is small (r=0.04/0.08).

Low priority
As the obtained p value is 0.28/0.25 we cannot reject the null-hypothesis. For the issues with low priority, the Mann-Whitney U test indicates that the lead time is equal for the technical debt issues (Median=226/223) and the security debt issues (Median=320).

Medium priority
As the obtained p value is 6.3E^{-5}/3.75E^{-9} we can reject the null-hypothesis and conclude that there is a difference in the lead time for technical debt and security debt. For the issues with medium priority, the Mann-Whitney U test indicates that the lead time is higher for security debt issues (Median=76) than technical debt issues (Median=73/58). The effect size is small (r=0.06/0.10).

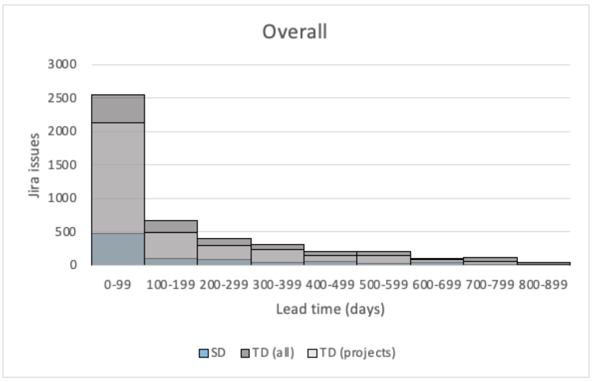
High priority
As the obtained p value is 0.001/0.013 we can reject the null-hypothesis and conclude that there is a difference in the lead time for technical debt and security debt. For the issues with high priority, the Mann-Whitney U test indicates that

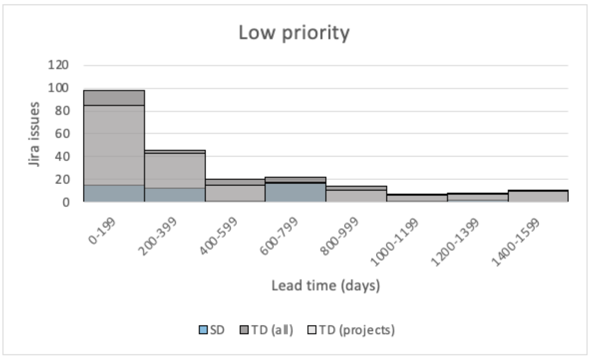
the lead time is higher for technical debt issues (Median=90/78) than security debt issues (Median=55). The effect size is small (r=0.14/0.11).

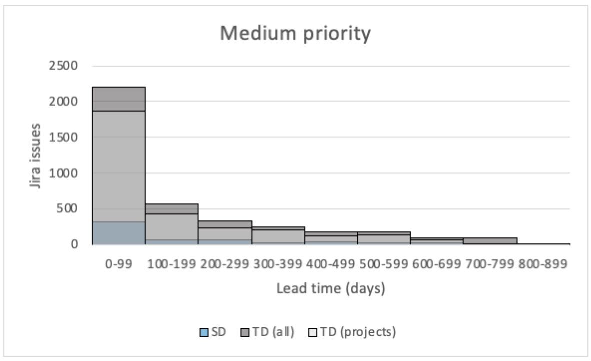
Critical priority

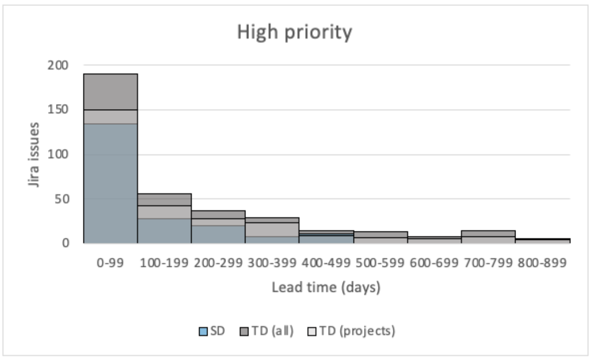
As the obtained p value is 0.045/0.028 we can reject the null-hypothesis and conclude that there is a difference in the lead time for technical debt and security debt. For the issues with critical priority, the Mann-Whitney U test indicates that the lead time is higher for security debt issues (Median=91) than technical debt issues (Median=39/31). The effect size is small (r=0.19/0.26).

The results from the Mann-Whitney U test if summarized in Table 4.3. This table shows that the results are not very different for the tests where all technical debt is included and where only the technical debt from the projects with security debt is included. At least, the difference does not impact if the result is significant or not.









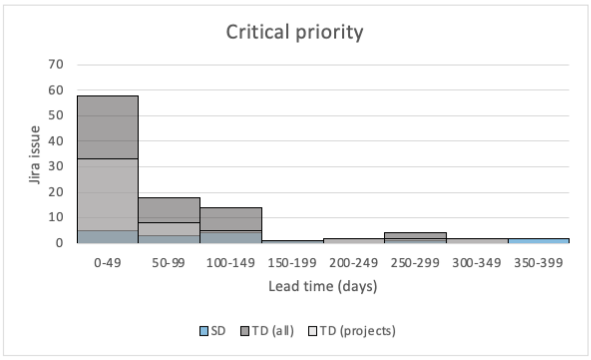


Figure 4.3: Overlapping histograms showing the distribution of the lead time for security debt (SD), all technical debt (TD all), and technical debt only from the projects that have security debt as well (TD projects).

Priority	Included issues	Significant difference	p value	Interpretation
Overall	TD (all) and SD	Yes	.002	**Technical debt** is fixed in **shorter time** than **security debt**
	TD (projects) and SD		$3.003E^{-8}$	
Low	TD (all) and SD	No	.28	There is **no difference** in the lead time for technical debt and security debt
	TD (projects) and SD		.248	
Medium	TD (all) and SD	Yes	$6.288E^{-5}$	**Technical debt** is fixed in **shorter time** than **security debt**
	TD (projects) and SD		$3.749E^{-9}$	
High	TD (all) and SD	Yes	.001	**Security debt** is fixed in **shorter time** than **technical debt**
	TD (projects) and SD		.013	
Critical	TD (all) and SD	Yes	.045	**Technical debt** is fixed in **shorter time** than **security debt**
	TD (projects) and SD		.028	

Table 4.3: Results from the Mann-Whitney U test

Chapter 5

Iterations

In this chapter I will describe how the artefact was created through two iterations. The iterations consists of step 3-5 in the Design Science Research process. As step 2 (defining objectives for solution) was the base from where I designed and developed the artefact I will describe this step first.

5.1 Defining the objectives for a solution

Before developing the artefact I discussed different ideas on how to solve the explicated problem together with the company stakeholders involved in our meetings. As the document analysis had already provided me with information about which data was available and the quantitative data analysis revealed insights about how the different teams have prioritized and fixed technical debt and security debt in practice we had good knowledge of the problem area. Based on this, the ideas were transformed into a set of functional requirements aiming to solve the problem. The six requirements presented in Table 5.1 was provided by the company stakeholders.

The explanation and motivation behind each requirement will be elaborated for in each of the visualizations below. The overarching goal for the artefact is that it shall serve as a dashboard of history keeping where the teams can see how they prioritized and fixed technical debt and security debt in the previous period and thus make better decisions regarding the prioritization of the debt in the future. These decisions can impact which debt is fixed and how fast they are fixed.

R1	Let teams know if they handle technical debt and security debt equally or not
R2	Let teams get information about how much time they use to fix technical debt and security debt
R3	Let teams get information about the time they use to fix issues compared to best team and average team in the company
R4	Let teams get information about the issues they have identified and fixed within the last period
R5	Let teams get information about the open security issues with highest priority
R6	Let teams get information about the open issues that have been in the system for the longest time

Table 5.1: Requirements proposed by the company stakeholders

5.2 First iteration

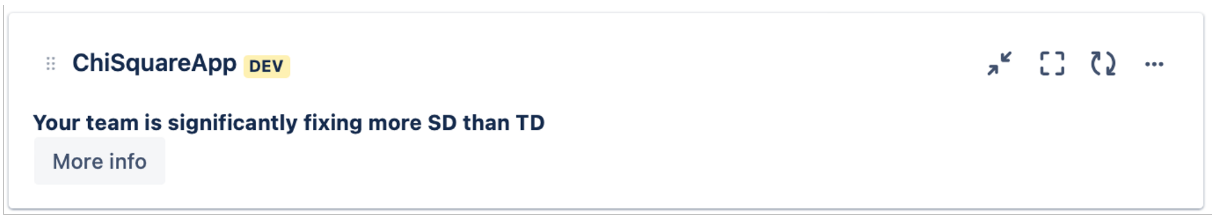
This first iteration includes the steps denoted with the letter *a* in my research process. The design and development phase were carried out based on the previously described requirements. As I have conducted the demonstration and evaluation at the same time, I will explain it together under the *Evaluation* sections.

The visualizations are created from the same data set that the statistical tests are performed on. Because the data set is extracted from the company's Jira backlog I decided to use Jira when creating the artefact. Before I could start to develop the visualizations I had to export data from TSD to Jira. I imported the five biggest projects in terms of highest amount of both technical debt issues and security debt issues. Before exporting the issues I replaced each issue's summary field with "summary" (as I didn't need this information for the visualizations) and labeled the issues as either "TD" (technical debt) or "SD" (security debt). This was done to keep the data as anonymous as possible and not display more information than needed.

5.2.1 Design and development

The visualizations shown below are created with data from one out of these five projects. I chose to use the middle-sized project to have a balance in the amount of presented information. The reason for using data from one project in the demonstration is that the artefact aims to serve as a dashboard for each of the teams. Since the teams within the company have a yearly assessment of the technical debt process, all of the visualizations are made with the time period of one year. As described in the background, I took inspiration from visualization techniques used in agile development when designing the artefact. The technology used for developing the visualizations is described in Chapter 6.

Visualization 1: The Chi-Square Test of Independence


ChiSquareApp DEV

Your team is significantly fixing more SD than TD

Figure 5.1: Visualization 1 - The Chi-Square Test of Independence

This visualization is created from the first requirement: **Let teams know if they handle technical debt and security debt equally or not.** The results from the quantitative data analysis revealed that security debt is fixed more often than technical debt. So, by letting teams get this knowledge they can see if they need to put more focus into the other debt type.

This visualization is based on the statistical test I performed when looking at the relation between issue type and resolution. The issues included here are the fixed and not fixed issues for both of the issue types within the last year. The visualization is configured just like the Chi-Square test of Independence described in section 3.4.1.

This visualization have three possible outputs:

- Your team is significantly fixing more technical debt than security debt
- Your team is significantly fixing more security debt than technical debt
- Your team is fixing technical debt and security debt equally

I added a "More info" button to show that it would be possible to get additional information if that was of interest for the teams.

Visualization 2: Created vs. resolved chart

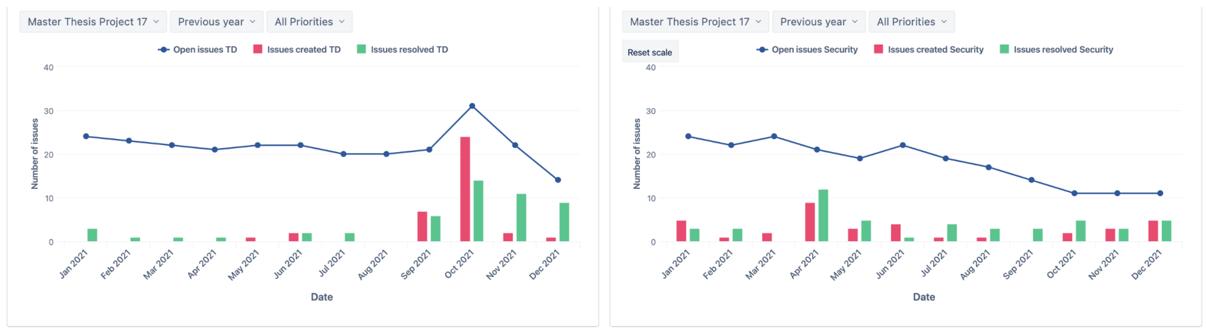


Figure 5.2: Visualization 2 - Created vs. resolved charts for technical debt and security debt

This visualization is created from the following requirement: **Let teams get information about the issues they have identified and fixed within the last period.** The first visualization only says if one of the issue types are fixed more often, it doesn't show how many issues that are identified and fixed by the teams. Because of this, it was discussed that showing the created and fixed issues would be helpful for the teams so that they can see if there are peaks (times where they identify or fix a lot of issues) in the graph. Additionally, by adding a line showing the open issues, the teams can see if they accumulate more or less issues over time.

This visualization is greatly inspired by the *created vs. resolved chart* that is an option to choose from in Jira. The difference is that I have included the open issues to show the trend of created and resolved issues over time.

This visualization includes one graph for technical debt issues (to the right) and one graph for the security debt issues (to the left). This is done so that the teams can compare how technical debt and security debt is being managed over time and if there are similarities/differences in how the two types are handled.

Visualization 3: Average lead time

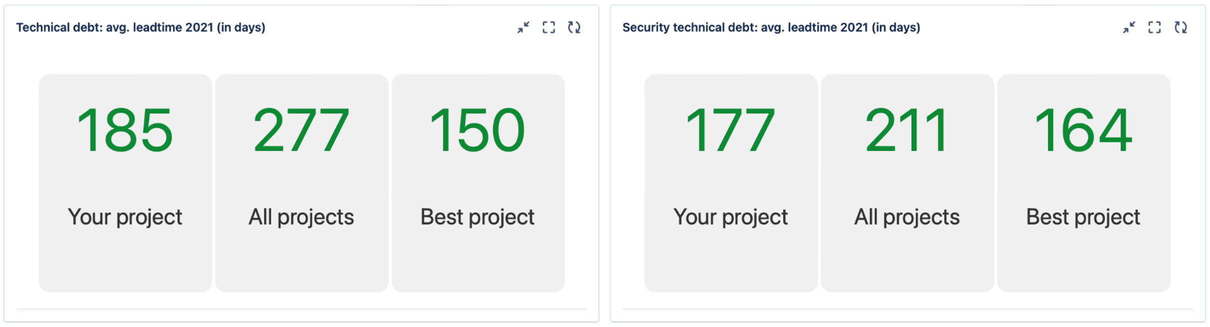

Technical debt: avg. leadtime 2021 (in days)
185
Your project
277
All projects
150
Best project
Security technical debt: avg. leadtime 2021 (in days)
177
Your project
211
All projects
164
Best project

Figure 5.3: Visualization 3 - Average lead time

This visualization is created from the following requirement: **Let teams get information about the time they use to fix issues compared to best team and average team in the company.** This visualization highlights an important metric for measuring productivity, namely the lead time. The reason for showing this is to let the teams know how much time they use to actually fix their issues. By letting teams get this information they can see how productive they are compared to other teams in the company. This can motivate them to take actions.

In addition, because the lead time is shown for both technical debt and security debt, the teams can see if they use much more or less time for one of these debt types. The visualization shows the average time (in days) used to fix an issue for their own project, for the company and for the best project. The average lead time for the technical debt items are shown to the left and the average lead time for the security debt issues are shown to the right.

The lead time is found in the same way as described earlier (created and resolved time stamps for each issue). This calculation might be misleading as the issue could have been untouched for half a year before it was started to work on. Additionally, the size of the issue could be a relevant metric to include here as it would require more time to fix a larger issue. But I only have access to the dates for when the issue was created and resolved. I don't have any information about the size or when the issue went from the status "To do" to "In progress" for example.

Visualization 4: Lead time chart

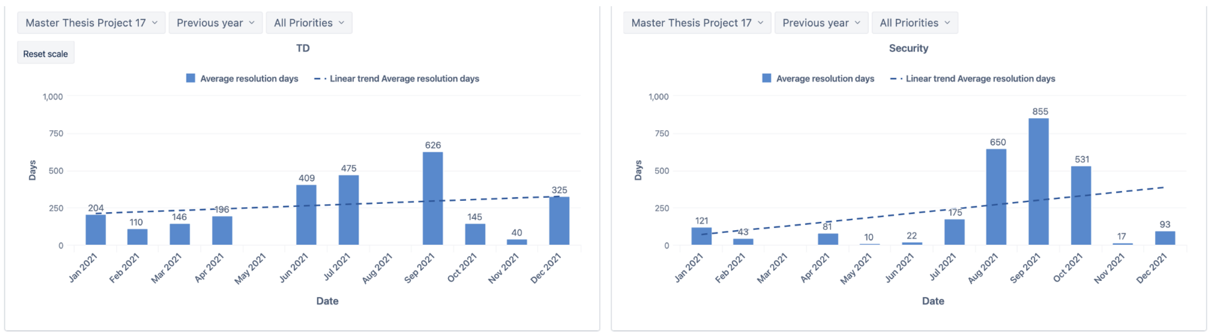


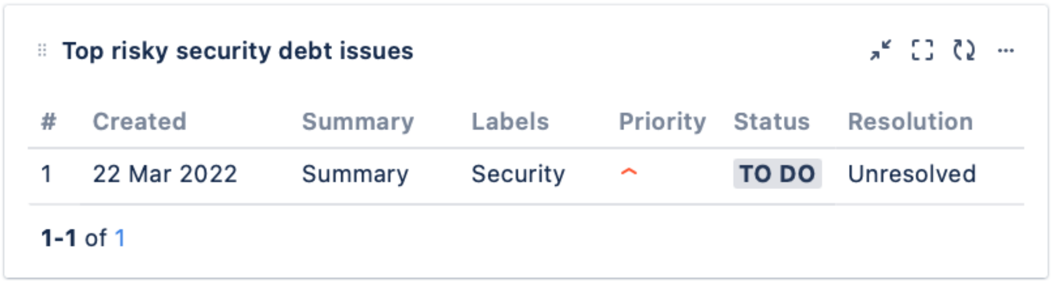
Figure 5.4: Visualization 4 - Lead time chart

This visualization is created from the following requirement: **Let teams get information about how much time they use to fix technical debt and security debt.** As already stated, the lead time is an important metric for measuring productivity. By letting teams get knowledge about how much time they have used on solving issues within the last period, they can see if they use more or less time than excepted and thus take actions if necessary.

The graph to the left shows the average lead time for technical debt while the graph to the right shows for security debt. As the previous visualization only shows the average lead time the last year and not how the lead time has changed throughout the year this graph is included. This is because it could reveal peaks (times where they fix a lot of old issues for example).

This was a chart that I was a bit unsure about how could best be visualized as the number of days between two months can be a bit confusing. For example one can see in the technical debt chart that it goes from 196 days in April 2021 to 409 days in June 2021. This can be confusing as it is not possible to add that many days in 2 months. But that only means that in June they have fixed older issues (issues created for a longer time ago).

Visualization 5: Top risky security debt issues


Top risky security debt issues

#	Created	Summary	Labels	Priority	Status	Resolution
1	22 Mar 2022	Summary	Security	^	TO DO	Unresolved

1-1 of 1

Figure 5.5: Visualization 5 - List of most risky security debt issues

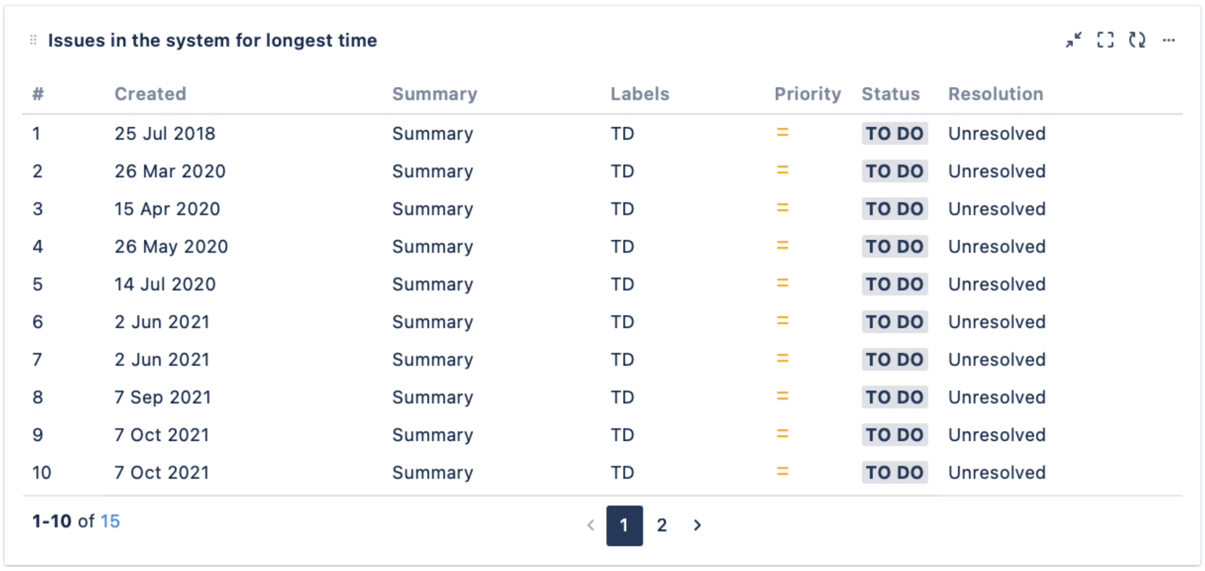
This visualization is created from the following requirement: **Let teams get information about the open security issues with highest priority.** As mentioned in the background, security debt is highly related to security risk. Because the security risk increases as time goes by (Rindell & Holvitie, 2019), it is important to highlight these kinds of issues. By displaying the open security issues to the teams it is easier to act upon them.

This visualization shows a list of the most risky issues, namely the unaddressed security debt issues. Only the issues with high and critical priority are included after discussions with company stakeholders.

Visualization 6: Oldest issues

This visualization is created from the following requirement: **Let teams get information about the open issues that have been in the system for the longest time.** By letting teams get information about the open issues that have been in the system for the longest time they can see if these issues are things they should worry about or not. It makes sense to display the oldest issues as these could represent candidates for refactoring or re-prioritization.

In contrast to the previous visualization, this list includes both technical debt and security debt issues. All of the priorities are included and the list is ordered by the created date in ascending order.



Issues in the system for longest time

#	Created	Summary	Labels	Priority	Status	Resolution
1	25 Jul 2018	Summary	TD	=	TO DO	Unresolved
2	26 Mar 2020	Summary	TD	=	TO DO	Unresolved
3	15 Apr 2020	Summary	TD	=	TO DO	Unresolved
4	26 May 2020	Summary	TD	=	TO DO	Unresolved
5	14 Jul 2020	Summary	TD	=	TO DO	Unresolved
6	2 Jun 2021	Summary	TD	=	TO DO	Unresolved
7	2 Jun 2021	Summary	TD	=	TO DO	Unresolved
8	7 Sep 2021	Summary	TD	=	TO DO	Unresolved
9	7 Oct 2021	Summary	TD	=	TO DO	Unresolved
10	7 Oct 2021	Summary	TD	=	TO DO	Unresolved

1-10 of 15

Figure 5.6: Visualization 6 - List of oldest issues

5.2.2 Evaluation

In this section I will present the feedback I got in the first evaluation. The feedback provided here was from a Chief Software Quality Engineer. As the findings from the statistical tests were a starting point for the rest of the thesis, it was natural to ask some questions regarding these results as well. This was also done in order to find out if these results were interesting for them to see and know about. Though, the main objective of conducting the interview were to evaluate the visualizations.

First, I will present the comments related to the findings from the quantitative data analysis. Then I will describe the feedback I got for each of the visualizations. In the end I will present some additional information that I found to be relevant to know of. The feedback provided in this interview led to refining the visualizations. These refinements are presented in the next section.

RQ1 How different is the amount of fixed issues for technical debt and security debt?

I explained that I have performed a statistical test to investigate the relation between issue type (technical debt or security debt) and whether or not the issue is fixed. Further I presented the results, which are described more detailed in section 4.1.

- **All priorities**: Significant result - security debt is more frequently fixed than technical debt
- **Low priority**: Not a significant result but security debt has a higher percentage of fixes than technical debt
- **Medium/high priority**: Significant result - security debt is more frequently fixed than technical debt
- **Critical priority**: Too few security issues that are not fixed to actually conduct the test

The participant found the results as expected and said that it would have been surprising if it was the other way around. Further, the participant told that they prioritize to fix everything that has to do with security before anything else.

The participant also pointed out why they thought it wasn't a big difference for the issues with low priority:

"I might have expected that everything that has a low priority, that there might not have been such a big difference because we encourage the teams to register most of their technical debt, and security debt too, and if there is something they are not going to fix maybe or maybe they will fix it a little far into the future, then we want them to just set the priority low and rather have it visible"
(first evaluation)

By doing it this way it is possible to act upon if the teams are accumulating a lot of issues.

RQ2 How different is the lead time for technical debt and security debt?

I explained that I have looked at the lead time for fixing the issues and described how I calculated the lead time. Further, I presented the results from the Mann-Whitney U test which is described more detailed in section **??**.

- **All priorities**: Significant result - the lead time for fixing a technical debt issue is less than for security debt
- **Low priority:** Not a significant result
- **Medium priority**: Significant result - the lead time for fixing technical debt is less than for security debt
- **High priority**: Significant result - the lead time for fixing security debt is less than for technical debt
- **Critical priority**: Significant result - the lead time for fixing technical debt is less than for security deb

The participant found the results a little bit surprising as they expected the security issues to be solved faster than the other technical debt issues. I explained that I did only have access to the two time stamps *created* and *resolved* and that I have not considered the size of the issues when calculating the lead time. Then the participant said that the size also could impact the time used to fix an issue:

"Some teams create technical debt, for example that they have to upgrade a very large version of a framework/library they use, and that can take many months to complete" (first evaluation)

The participant told that an important question related to the lead time is how the different teams prioritize the issues in order to perhaps make it visible or not. Additionally, the participant mentioned that some of the teams might not even tag the issues that are technical debt:

"You can tag it as that this is a bit of a bigger project even though it is basically technical debt" (first evaluation)

Visualization 1: Chi-Square app

The participant expressed that this first visualization was useful for the teams so that they could see how much focus they have on technical debt and security debt during a period. Further, the participant said that some additional information should be presented as well:

"... we have looked at teams and if we present them with a statement like yours, they are also interested in the data behind the statement" (first evaluation)

Therefore, it was proposed that showing how many issues that were fixed for both of the issue types would be an easy way to see the data behind the statement. Additionally, the participant proposed to present a more dynamic app by letting the teams choose which time period they would like to see the visualization for in order to have a more continuous overview. It was suggested that the teams should be able to filter the last month, last three months, last six months and the last year.

Visualization 2: Created vs. resolved

This visualization was perceived as very useful to the participant. This was because it could reveal if the teams are working continuously with the issues or not. The participant told that it is seen as important (to the company and management) to have a good culture of fixing technical debt continuously. Further, the participant pointed out the importance of not only looking at the number of created and resolved issues:

"... because we have seen a few times, if we look at how many issues have been solved and created in the last 12 months, we get a number, but has it been done one week before we go into review because then they realized they have to focus or they will get some points?" (first evaluation)

The participant were curious if the issues that were created and fixed were only low priority issues or if there were critical issues as well. Based on this, it was proposed to have a visualization showing more "deeply" the issues that are created and fixed.

Visualization 3: Average lead time

This visualization was perceived as interesting and the participant pointed out that it could be useful for the teams in terms of getting an impression of how much time they use to resolve an issue. Further, the participant were a bit unsure of how it would be to compare the lead time between different teams in the company. This was because the teams have different ways of coming through the development process and different requirements for their security, so some places things will take longer or go quicker.

Immediately, the participant thought that there should be possible to filter based on the different priorities:

"Guessing that these 185 days sounds like a lot, but it's probably not for critical or high issues, but maybe more on low and medium" (first evaluation)

Visualization 4: Lead time graph

The participant told that they have had a wish to see how long time the different teams have used to solve a Jira case before (not necessarily technical debt). Then they have seen that if the team addresses an issue that was created three years ago there comes a huge spike in the graph. The participant emphasized this by looking at the graph for technical debt in September 2021:

"It could be that they have solved 10 different cases that are so so old and then it accumulates in 600 days, or they have solved one case that is 620 days and two other cases that are 2 days old" (first evaluation)

Based on this, the participant suggested to think of another way to visualize this as the teams could find it hard to interpret this graph. In conclusion, the participant found it very interesting for the teams to see if the issues have been in the system for a long time. Though, the participant proposed, at least for the team perspective, to show a list of the fixed items with a column counting the lead time.

Visualization 5 and 6: Lists of most risky issues and oldest issues

As these two visualizations serves the same purpose of showing the most important issues, they were discussed at the same time. According to the participant, these visualizations are definitely perceived as useful for the different teams. It was stated that this kind of filtering, both for the oldest issues but also for the priorities were very useful.

Additional relevant information

As an introduction to the interview I asked some questions about how the teams are logging their technical debt and security debt. This was done to find out how they label the different issues and if they have a practice of registering most of the issues in Jira or not. The participant informed me that some teams are very strict on technical debt in terms of almost fixing the debt before they register it in Jira. Thus, the participant told me to keep in mind that there are many teams that are actually really good at managing technical debt even if they don't document it in any ways:

"The team maybe have some code review sessions where they find technical debt, and then they fix it immediately because they know that if they register a case in Jira they have to close it in so so many hours and then they choose to not do it" (first evaluation)

Generally concerning visualizations, the participant expressed that they don't like it when a graph shows too much metrics/information at the same time. Although there can be a lot of good information in one graph, the participant told that they like to keep it to show one graph per thing they look at. In this way there is not too much information to process at one time.

When it comes to additional visualizations that the teams could benefit from, the participant mentioned three fields they use in relation to technical debt in Jira: risk impact, risk likelihood and severity. The participant expressed that these fields could be interesting to show to the teams in any way, thus it was said that the priority field (currently used in my visualizations) also works for the same purpose. As the data set I have looked at only holds information related to the priority, the three other mentioned fields will not be used in any of the visualizations.

5.3 Second iteration

This iteration is conducted after the first iteration and denoted with the letter *b* in my research process. The feedback given in the interview in the first iteration were used to refine the visualizations. This will be described below. In the end I will go through how these refined visualizations are demonstrated and evaluated.

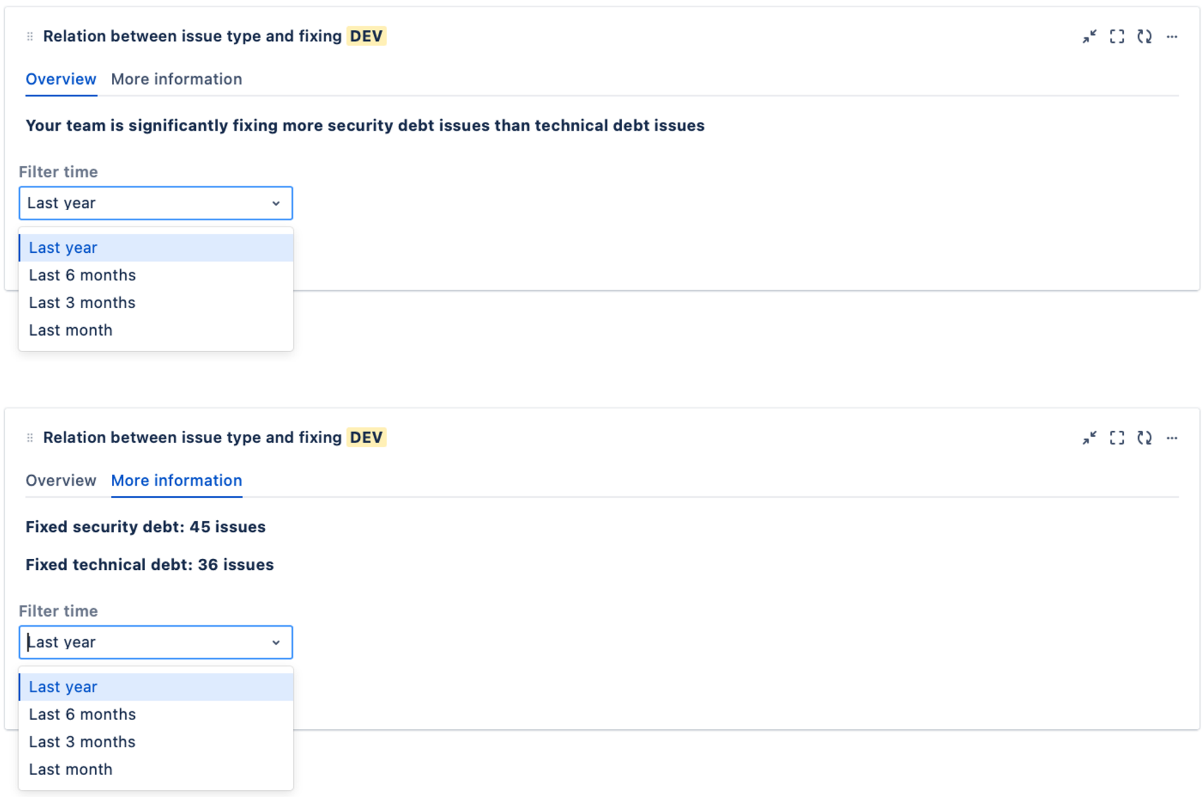
5.3.1 Refining the visualizations

The visualizations are using the same data (from one project) as in the first iteration. For the demonstration in this step, all of the visualizations were put together in a dashboard in Jira. The entire dashboard is shown in chapter 6.2. Below I will go through how each visualization has been refined.

Visualization 1: Chi-Square app

Because the feedback related to this visualization concerned more information in terms of the knowing which data lays behind the statement, and the possibility of filtering time period, these functionalities were added to the visualization.

The visualization has now an "Overview" tab that displays if the team is focusing more on one of the issue types or not. Furthermore, there is a "More information" tab that displays the number of fixed issues during the chosen time-period. Lastly, the viewer can choose to filter between the following time-periods: last year, last six months, last three months, and the last month.


Relation between issue type and fixing DEV

Overview | More information

Your team is significantly fixing more security debt issues than technical debt issues

Filter time: Last year

- Last year
- Last 6 months
- Last 3 months
- Last month

Relation between issue type and fixing DEV

Overview | More information

Fixed security debt: 45 issues

Fixed technical debt: 36 issues

Filter time: Last year

- Last year
- Last 6 months
- Last 3 months
- Last month

Figure 5.7: Refined visualization 1 - The Chi-Square Test of Independence

Visualization 2: Created vs. resolved by priority

The feedback related to this visualization focused on the importance of showing the distribution of priorities for the created and resolved issues. The visualization is now refined to show the issue's priority. The created issues are marked with a degree of red color while the fixed issues are marked with a degree of green color. The darker the color is, the higher is the priority.

As the participant pointed out that it was essential to not show to much information in the same graph I decided to not include the open issues. Further, the functionality of choosing between the last year, last six months, last three months and last month is also added.

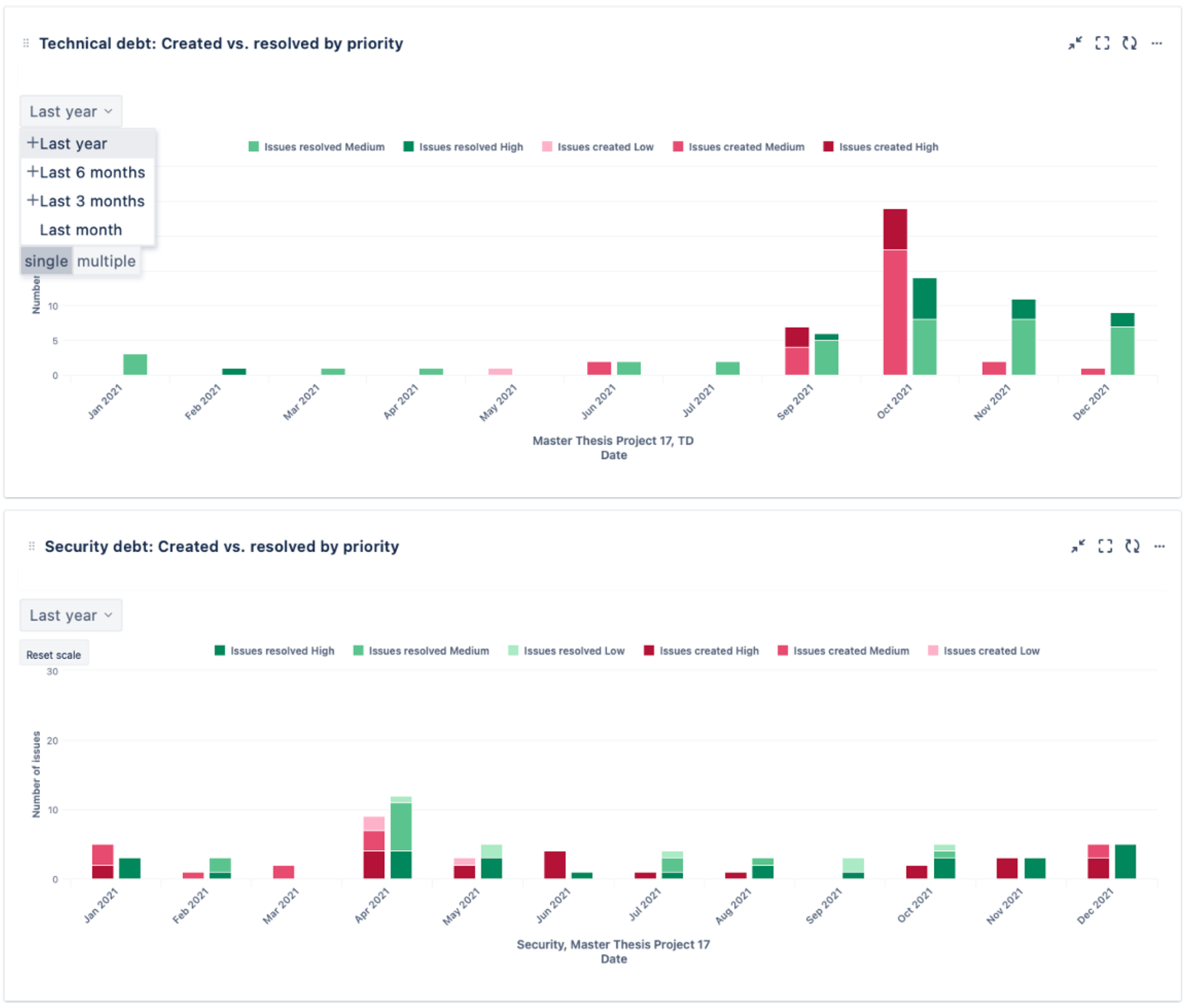
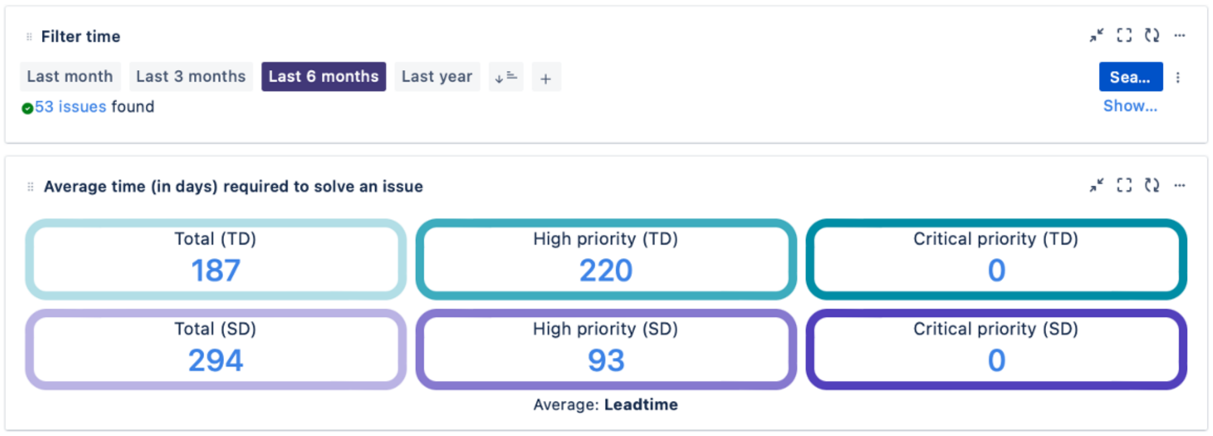


Figure 5.8: Refined visualization 2 - Created vs. resolved charts by priority

Visualization 3: Lead time

The feedback related to this visualization concerned seeing the lead time for the different priorities. Because the issues with low, or even medium priority are not considered very important, I have chosen to show the lead time in total (for all priorities) and the lead time for the issues with high and critical priority. Additionally, I did remove the comparison of lead time between the best team and all teams in the company.

Visualization 4 (in the first iteration) is also related to the lead time. The feedback regarding this visualization were that it could be confusing to interpret it. It was



Average time (in days) required to solve an issue

Total (TD) 187	High priority (TD) 220	Critical priority (TD) 0
Total (SD) 294	High priority (SD) 93	Critical priority (SD) 0

Average: **Leadtime**

Figure 5.9: Refined visualization 3 - Average lead time

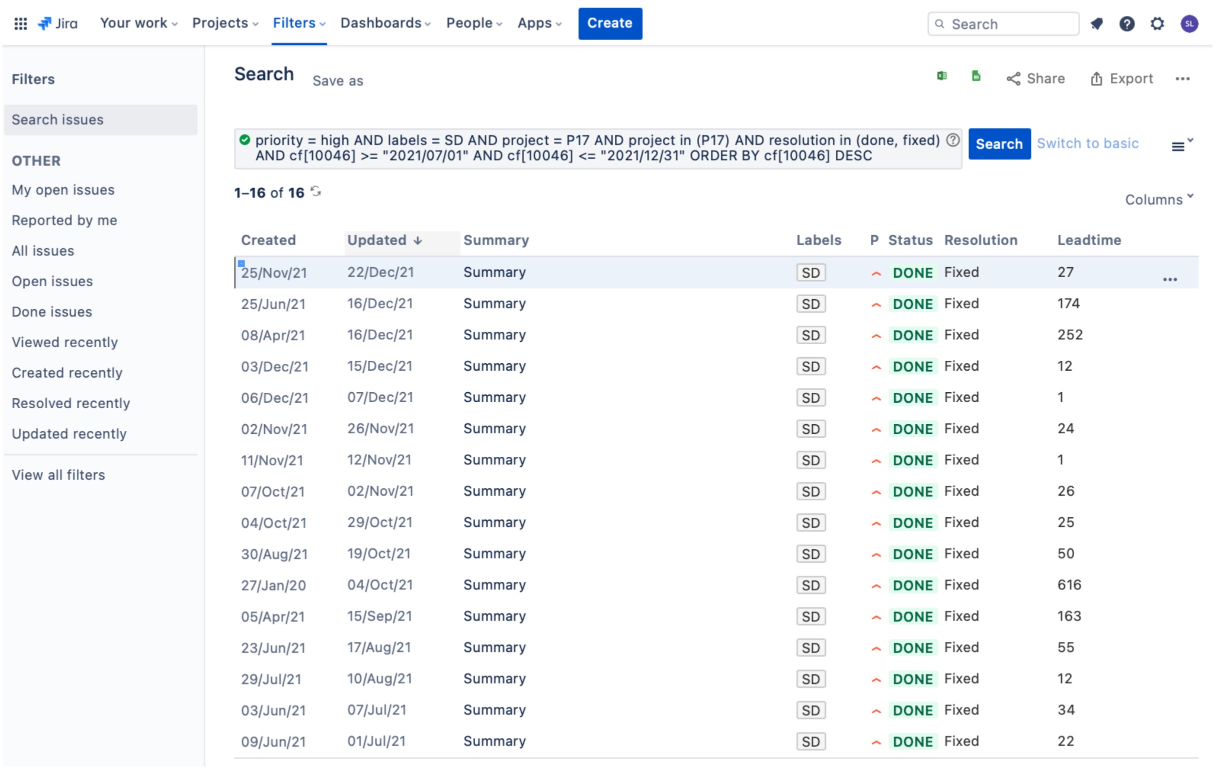
suggested to show a list of the fixed items with a column counting the number of days since they were created.

I have merged the these two visualizations (visualization 3 and 4 from the first iteration) into one visual representation. This refined visualization shows the average lead time (in days) for a chosen time period. The viewer can choose between the same time periods as in the two previous; last year, last 6 months, last 3 months and last month. In order to also include the list suggested by the participant in the initial interview, it is now possible to click at the boxes to open a new tab showing the list of these included issues.

By for example clicking the high priority security debt issues while having chosen the last 6 months, the list shown in Figure 5.11 will open in a new tab. As seen from this figure, the issues that are resolved between July 2021 and December 2021 are included with a separate column showing the lead time for each item.

Visualization 4: Important issues

As visualization 5 (risky issues) and visualization 6 (oldest issues) from the first iteration were discussed at the same time I decided to merge these into one visualization concerning the most important issues. From having them in two separate lists I have now made one visualization that only displays the number of issues in each category. The risky issues are still the unaddressed security issues of high or critical priority, and the other box shows the unaddressed issues that are older


Search

priority = high AND labels = SD AND project = P17 AND project in (P17) AND resolution in (done, fixed) AND cf[10046] >= "2021/07/01" AND cf[10046] <= "2021/12/31" ORDER BY cf[10046] DESC

1–16 of 16

Created	Updated	Summary	Labels	P	Status	Resolution	Leadtime
25/Nov/21	22/Dec/21	Summary	SD		DONE	Fixed	27
25/Jun/21	16/Dec/21	Summary	SD		DONE	Fixed	174
08/Apr/21	16/Dec/21	Summary	SD		DONE	Fixed	252
03/Dec/21	15/Dec/21	Summary	SD		DONE	Fixed	12
06/Dec/21	07/Dec/21	Summary	SD		DONE	Fixed	1
02/Nov/21	26/Nov/21	Summary	SD		DONE	Fixed	24
11/Nov/21	12/Nov/21	Summary	SD		DONE	Fixed	1
07/Oct/21	02/Nov/21	Summary	SD		DONE	Fixed	26
04/Oct/21	29/Oct/21	Summary	SD		DONE	Fixed	25
30/Aug/21	19/Oct/21	Summary	SD		DONE	Fixed	50
27/Jan/20	04/Oct/21	Summary	SD		DONE	Fixed	616
05/Apr/21	15/Sep/21	Summary	SD		DONE	Fixed	163
23/Jun/21	17/Aug/21	Summary	SD		DONE	Fixed	55
29/Jul/21	10/Aug/21	Summary	SD		DONE	Fixed	12
03/Jun/21	07/Jul/21	Summary	SD		DONE	Fixed	34
09/Jun/21	01/Jul/21	Summary	SD		DONE	Fixed	22

Figure 5.10: List shown in a new tab when clicking on the *High priority (SD)* box (for the last 6 months)

than a year. I chose to only display the number because by clicking the box, the viewer will get another tab showing a list of the included issues (just like in the previous visualization).

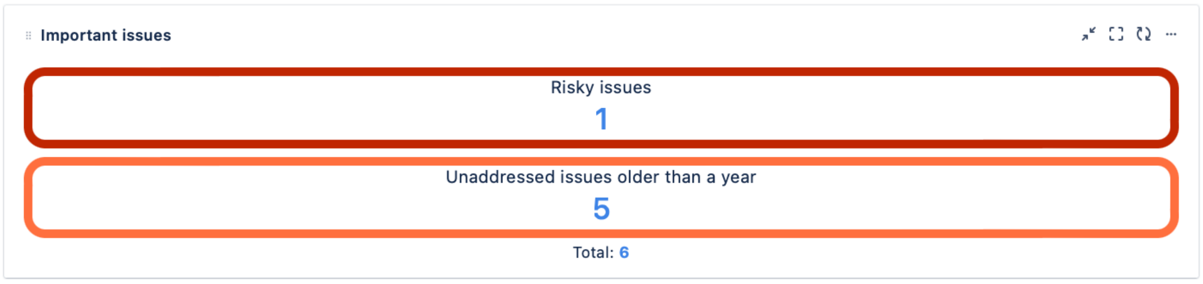


Figure 5.11: Refined visualization 4 - Important issues

5.3.2 Evaluation

In this section I will present the feedback provided in the last evaluation. The participants have different roles and are from five different teams/projects. There was one service architect, one development manager, two back-end developers (from the same team), one project manager, and four architects (from same project).

The evaluation criteria for the second iteration were to find out about the **understandability** and **usefulness** of the visualizations. I will go through the results regarding this criteria for each of the visualizations below. Some of the participants also suggested improvements. These improvements will be described below as well.

To be able to distinguish between the information provided by the participants in the different interviews I'm referring to the interviews as they are described in Table 3.2.

Visualization 1: Relation between issue type and fixing

Understandability
All of the participants found it easy to interpret this visualization. One of the participants said:

"This team finds security debt more important than technical debt is basically what it's saying" (A1)

Before I demonstrated the visualizations to the participants, I shortly explained what I had done in the project before creating the visualizations. This was done to establish the context for how the visualizations were developed. As this visualization reflects one of the statistical tests that I have performed (and told about), that can have impacted the participants understanding here.

Usefulness
Regarding the usefulness of this visualization it was said in four of the interviews that this was interesting insights to have about a team/project. Participant D1

pointed out that this visualization was a good first view to get an overview of security debt and technical debt. Further, this participant said:

"Well I think it is interesting insights, because first of all it gives you kind of heads up to put some focus on the fact that there is a difference on the technical debt and security debt, which we haven't really always been as clear about focusing on" (D1)

Moreover, this participant told that they had biweekly meetings where they discussed technical debt and said that they would have used this visualization in this meeting.

Participant B1 pointed out that the stakeholders could find it interesting to know of this as well as getting an overview of the issues that have been completed during the last period.

Improvements
Some of the participants suggested improvements for the visualization. Participant B1 would like to have a weekly option in addition to the other options. Participant D1 also suggested a more granular time filtering in order to configure it to fit their planning periods (two weeks at a time).

Further, participant E1 proposed to display some proportion, for example if you fix twice as many security issues than technical debt issues. This was because you could then define a metric that you aim for. Participant E2 also suggested to see some more statistics (like story points or time spent on the issues).

Participant B1 asked if this visualization were just for Jira as the participant thought it could be pretty cool if the visualization could have a hyperlink to the issues that are filtered. Lastly, participant in D1 said that it could be interesting to see the different priorities for the issues that are fixed as this have a great impact on which items they fix first.

Visualization 2: Created vs. resolved

Understandability
Before I asked questions related to each of the visualizations in the dashboard I

let the participants have a look and asked them where they were looking. This visualization were mentioned by several of the participants as the first visualization that caught their eyes. Regarding the understandability, all of the participants found it easy to interpret these graphs. One of the participants said:

"These graphs are simple views to get a good understanding on how you are actually performing" (B1)

In addition, several of the participants told me different things they saw in the two graphs and discussed the created and resolved issues.

Usefulness

The usefulness of these graphs were dependent on the amount of issues in their backlog. Participant A1 told that because they have that few issues they wouldn't look at this visualization monthly or weekly. Further, the participant said:

"If you could sometimes look back for one or two years then you might see some discrepancies that you want to improve" (A1)

Participant B1 said that one of the key performance indicators (KPIs) that they have is how good they are at resolving the issues coming in. Moreover, the participant told that these graphs would be very useful too see if they are actually delivering on their KPIs and to identify if they needed to have more focus on cleaning out some of their technical debt. In addition, the participant told that these graphs could be useful for creating presentations internally (for the rest of the team) to let them see that they are making progress and doing better.

Participant D1 said that it would be useful in the sense of revealing how the prioritization has been in the previous period, especially helpful to become more aware of the issues that have been re-prioritized without them actually updating the risk severity. Moreover, the participant told that it would be relevant to look at during their biweekly planning meetings as well as on a higher level of planning (for example quarterly plannings). By looking at these graphs they could get more detailed insights into how they have actually worked with technical debt for the last quarter and whether they should be more aware of changing their priorities for the coming quarter.

In addition, most of the participants mentioned that they really liked that the graphs offered another dimension than an ordinary created vs. resolved chart, namely the priorities. Two of the participants said:

"I think one of the interesting things is the priorities, because even if we have a technical debt policy stating that we should always prioritize technical debt with the highest priority first, that is not always the case in practice" (D1)

"I like it, I like that they have another dimension based on the priority" (E2)

Improvements
Some of the participant suggested improvements for these graphs. Participant E3 would like to include a goal/threshold with how many tickets the team should tackle per month so that they could see if they over-deliver or under-deliver. In one of the interviews I asked the participant if they would have included a goal/threshold in these graphs where I got the answer:

"No, we would not use that. As I mentioned, we have the KPIs, for example that we are not gonna have more than 10 bugs open, so to be able to see this it is already more than enough for us actually" (B1)

Participant D1 suggested to include some trend lines in the graphs. This was also suggested by the participants E1, E2 and E3 to be able to see if they were trending up or down or holding steady.

In addition to the time periods available in the graphs, participant D1 would also like to be able to filter more granular. This participant also thought that it could be easier to compare the two graphs if they were merged into one graph, i.e., to show the technical debt issues and security debt issues in the same graph.

Visualization 3: Average lead time

Understandability
This visualization were not the easiest to interpret right away. Participant E1 asked what these numbers meant. Otherwise, the questions regarding this visualization were mainly concerned with if the numbers were in minutes or days.

Most of the participants told that they had never used the lead time metric in relation to technical debt before. Participant A1 said that they used the lead time to measure the deployment from merging to master production but didn't have an idea of technical debt and security debt lead time.

Although, after looking at the visualization for some time, the participants told me different things about the visualization:

"This project takes about 6 months to resolve technical debt issues" (A1)

"... we can also see that it takes more than twice less time to solve a security debt of high priority than for technical debt" (C2)

Usefulness
Regarding the usefulness, participant A1 found that the created vs. resolved charts revealed more than this visualization on how they work with technical debt and security debt over time. However, the participant would have used the list of resolved issues with a column showing the lead time.

Participant D1 also pointed out that they would imagine that this lead time would need to be looked at in a context of a longer time frame (more high-level planning). The participant thought that this visualization could be valuable to look at in the quarterly plannings to see if the lead time is too high based on what they would expect:

"Then maybe we would have to put some effort into having technical debt pay down days. It could affect the planning in that sense" (D1)

Additionally, participant D1 said that with a year to evaluate, these numbers could be really good to actually set some goal metrics in terms of where they want to be to be a high performing software development team.

Participant E3 told that this visualization would be useful to define deadlines. Participant B1 said that the lead time is an important metric. Further, the participant emphasized who would have used the visualization:

"Well, I think every team would have actually used it because we have KPIs in place for lead time and it makes reporting so much easier, and also for the teams to have an idea on how they are performing compared to out KPIs" (B1)

After demonstrating the visualization and showing that the boxes could be clicked on to get more details, participant B1 said that this was something that could be used on a retrospective meeting on all issues if something has been taking longer than it should have.

Participant C1 said that the Service owner could have used this visualization to see how long time they have used to solve a technical debt issue. Further, this participant said that this visualization were also useful for them:

"If we see that it takes three months to solve a high priority security issue, we should re-prioritize a bit" (C1)

Improvements

Some of the participants suggested improvements for this visualization. Participant E1 would have liked to see something with a trend of time:

"We have the actual numbers at the moment, but maybe something can be done to visualize if you are declining on lead time, maybe based on priority" (E1)

Participant D1 would also like to see if they were resolving technical debt tickets faster or slower over time. In addition, this participant suggested to show the variances (maximum and minimum values) as well:

"... but actually for one of the high priority issues we have used 900 days and one of them was resolved within 1 day, why is there such a difference? Especially if they are within the same priority" (D1)

Visualization 4: Important issues

Understandability

Some of the participants mentioned this visualization as the visualization they firstly looked at. Although all of the participants understood that this visualization reflected the most important issues in the system, it was not as clear which issues were considered important. Participant C1 asked what the risky issues meant.

Additionally, there was quite a difference in the participants thoughts about when the issue is considered *old*. Participant A1 said that if the issues were of medium and especially low priority, it didn't matter if the issue were years old. Participant D1 said that if they had technical debt that is older than a year they would want to know about it. Further, this participant said that because their team is quite small and their projects as well, they attempt to not have technical debt older than a year.

Usefulness
I asked the participants what was the most important things to see in order to make the planning and prioritization of technical debt and security debt easier. Participant A1 mentioned two things where one of those were a list of the most important issues.

Participant B1 told that they really liked the way I was highlighting the important issues and that they thought it could add a lot of value. Further, the participant said that if it was possible to see the issues related to the KPIs they have defined that could be really interesting for the service owners to see (and perhaps tech lead of each team to create some awareness). In addition, product owners and stakeholders could benefit from looking at this visualization before sprint planning. Regarding when the visualization would be used, the participant said that:

"For myself it would probably be more than once a week that I look into this" (B1)

Additionally, the participant really liked the possibility of drilling down into the details:

"From this view as well it might be that you look at it today and you see that we have one risky issue and then goes into the details, and tomorrow when you log in you still have one issue it's probably the same so you don't bother checking the details" (B1)

Participant E1 said that they would have used this visualization themselves as a tool to help prioritize the debt. The participant said that it would be especially useful in relation to the prioritizing activity if they had some calculations in there that took into account the factors for the security index (index used by all projects in the company):

"For example when we have a bug bounty ticket we have 90 days to resolve it, otherwise it [the security index] starts counting against us" (E1)

Because participant D1 don't have that many issues in their backlog, the participant thought that they would benefit more from seeing an overview of all their tickets. The participant pointed out that this would probably depend on the team and the size of the project:

"For our sake, with the process that we are used to now, where we do this review of all the tickets every other week, we don't go into details for all of them but we can shuffle through them and have an overview. Because we have the time to do that and we don't have that many tickets I think that is preferable. To see, maybe just highlighting the important issues, that could maybe for us lead to kind of being less aware than we currently are" (D1)

Improvements

Some of the participants suggested improvement related to this visualization. Participant B1 suggested to define some KPIs of how long you expect the most major issues to live in Jira, and then show the visualization based on that. Participant C1 were curious if there was possible to add a functionality of sending a notification by mail:

"For example if you create a security issue of high priority you can get a notification in mailing list or something" (C1)

Participant C2 emphasized this by saying that it would be nice to get notification by mail for the issues with high and critical priority. Participant E3 would like to add another dimension, i.e., make it of the level of severity and also likelihood:

"Because sometimes a small issue that everybody is facing might be more important than a super critical issue that rarely happens" (E3)

Table 5.2 below shows a summarized version of the results provided in this second evaluation.

	Understandability	Usefulness	Improvements
Chi-Square	• Easy to interpret • The fact that I told about the Chi-Square test could have an impact	• Interesting insights • Overview of technical debt and security debt • Can reveal if there is a difference which the teams haven't focused on ***When***: • Bi-weekly planning meetings • Before each release	• More granular time filtering • Display proportion • Hyperlink to issues • See distribution of priority
Created vs. resolved by priority	• Simple views • Easy to interpret	• Looking back for one or two years can reveal discrepancies to improve • Useful to see if they are delivering on their KPIs • Useful for creating presentations internally • Useful for revealing how the prioritization has been in the previous period ***When***: • Bi-weekly planning meetings • Quarterly planning	• Include goal/threshold • Include trend lines • More granular time filtering • Merge graphs into one (to easier compare)

Table 5.2: Summarized results for the two first visualizations from the second evaluation

	Understandability	Usefulness	Improvements
Lead time	• Not the easiest to interpret • Most of participants had never used lead time in relation to technical debt before • After looking for some time, the participants told me several things they saw from the visualization	• Can reveal if the lead time is higher than expected • Useful for setting goal metrics (yearly evaluation) • Useful to define deadlines • Useful for teams to have an idea on how they perform according to KPIs • Useful to see if issues should be re-prioritized ***When***: • High-level planning • Quarterly planning • Retrospective meeting	• See trend over time • Show variances (min and max values)
Important issues	• Understood that it reflects the most important issues • Not clear which issues were considered important • Different views on when issues are considered old	• Useful as a tool to help prioritize/re-prioritize • Useful for planning which issues to fix ***When***: • Before sprint planning • Regularly	• Configure it to their considerations of important issues (security index, KPIs) • Add e-mail notification • Include severity and likelihood

Table 5.2 Continued: Summarized results for the two last visualizations from the second evaluation

Chapter 6

Artefact

In this chapter I will describe the technology I have used in order to develop the artefact. In addition, I will go through the main functionalities that the artefact provides.

6.1 Technology

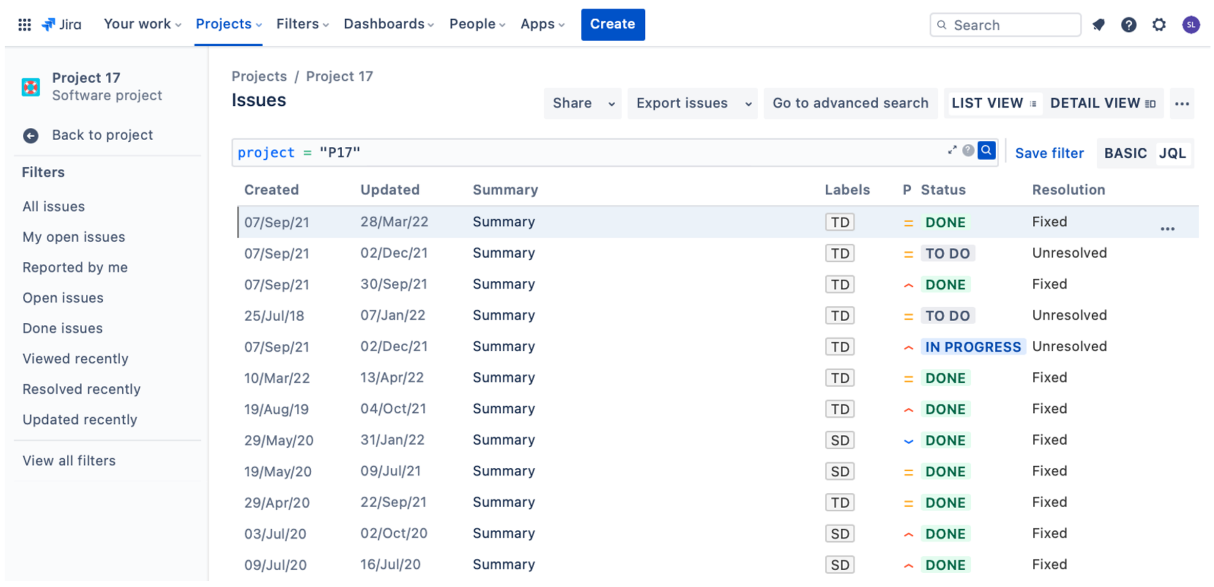
This section presents the technology used to develop the visualizations. I chose to use Jira [1] because the data set I investigated was extracted from the company's Jira backlog. The stakeholders that I have interviewed have a good knowledge of how Jira works which I think that is an advantage when demonstrating the artefact. Additionally, the visualizations can easily be adopted by the company if they want to.

6.1.1 Jira Cloud Platform

As described above, I chose to use the Jira platform to develop my artefact. I imported the data from the five largest projects into Jira by making a CSV file. Below is a figure showing the backlog of technical debt and security debt issues

[1] https://www.atlassian.com/software/jira

for one of the projects. This figure also shows which fields that exists in the data set I have access to.



```
project = "P17"
```

Created	Updated	Summary	Labels	P	Status	Resolution
07/Sep/21	28/Mar/22	Summary	TD	=	DONE	Fixed
07/Sep/21	02/Dec/21	Summary	TD	=	TO DO	Unresolved
07/Sep/21	30/Sep/21	Summary	TD	^	DONE	Fixed
25/Jul/18	07/Jan/22	Summary	TD	=	TO DO	Unresolved
07/Sep/21	02/Dec/21	Summary	TD	^	IN PROGRESS	Unresolved
10/Mar/22	13/Apr/22	Summary	TD	=	DONE	Fixed
19/Aug/19	04/Oct/21	Summary	TD	^	DONE	Fixed
29/May/20	31/Jan/22	Summary	SD	⌄	DONE	Fixed
19/May/20	09/Jul/21	Summary	SD	=	DONE	Fixed
29/Apr/20	22/Sep/21	Summary	TD	=	DONE	Fixed
03/Jul/20	02/Oct/20	Summary	SD	^	DONE	Fixed
09/Jul/20	16/Jul/20	Summary	SD	^	DONE	Fixed

Figure 6.1: Example backlog in Jira showing the technical debt and security debt issues for one of the projects.

Then, with the imported data I started to design and create the different visualizations in the dashboard functionality in Jira. When I had made some different visualizations I showed it to my supervisors and company stakeholders in our weekly meeting to get some feedback and input.

6.1.2 EazyBI

In order to create the visualization showing the created and resolved issues by priority I used EazyBI [2]. The reason for using this application was that it allowed the representation of both bars and lines within the same graph (which is not possible in the Custom Charts for Jira). EazyBI is an add-on available in the Atlassian Marketplace [3]. This app lets you visualize and analyse your data by creating custom Jira reports, charts and dashboards.

[2]https://eazybi.com
[3]https://marketplace.atlassian.com

6.1.3 Forge app

In order to visualize the relation between issue type and resolution (Chi-Square test of independence) I had to create my own app as this could not be done in an existing app/add-on available in Atlassian Marketplace. I chose to create a Forge app because this is the recommended choice by Atlassian [4]. Forge apps are fully hosted by Atlassian and Atlassian automatically creates the development, staging, and production environment (Atlassian, 2022a).

The Forge app I created calls a remote resource, namely the Jira REST API. I used the @forge/api package to make REST calls to an authenticated Jira endpoint. By using this package I got access to the Jira issues in the form of an array.

To create a dynamic and interactive interface for my app's frontend I used the UI kit provided by Atlassian. The UI Kit is made up of three main concepts: components, hooks and event handlers (Atlassian, 2022b). For my app, I used the *DashboardGadget* and *DashboardGadgetEdit* component as well as a *Select* form. The *Select* form were used so that the viewers could filter time.

6.1.4 Custom Chart for Jira

For the visualization of lead time and important issues I used the *Custom Chart for Jira* app because this app offers custom reporting directly inside the Jira dashboard. I customized the dashboard gadgets using the Jira Query Language (JQL). An example of a JQL query (for including security issues with high or critical priority) will be shown below:

labels = "SD" AND priority IN ("High", "Critical")

Additionally, for the lead time visualization I used a separate dashboard gadget called *Simple Search* so that the dashboard viewers could filter on which time period they would like to see the visualizations for. This search gadget were connected to the other gadget so that they showed the same data.

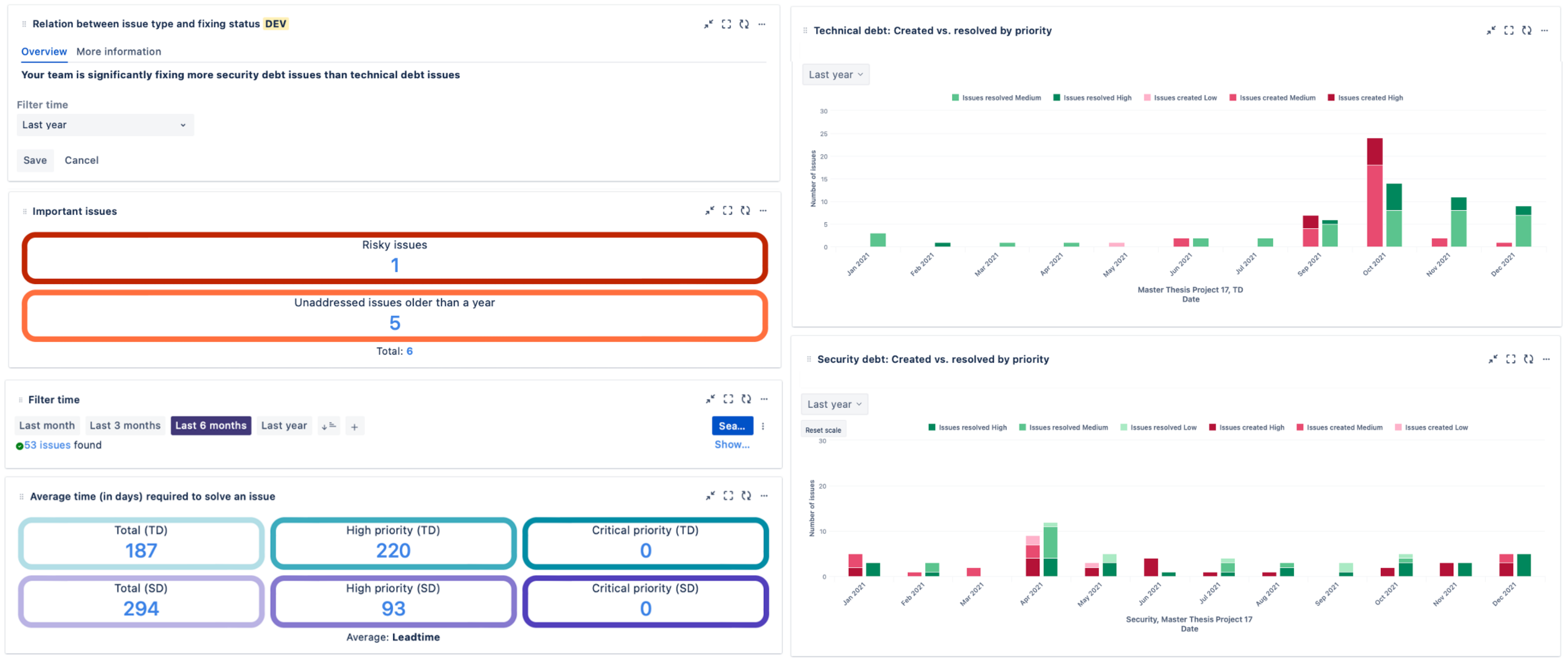
[4]https://atlassian.com

6.2 Dashboard

The artefact I have created aims to serve as a dashboard for the teams to get an overview and understanding of how they have managed/prioritized technical debt and security debt in the previous period, as well as get an insight into the most important issues that exists in their project. The final dashboard consists of four different visualizations. These visualizations are described in section 5.3.1. The entire dashboard is shown in Figure 6.2.

As the artefact is implemented in Jira, all functionalities that exists in Jira can be used in parallel to the created dashboard. Though, this is out of scope for this thesis so I will not go further into this. Moreover, I will shortly explain how the viewer can interact with the dashboard.

For all of the relevant visualizations, the viewer can **filter time** by choosing either to view the last year, last 6 months, last 3 months or last month. For the visualizations of important issues and lead time it is possible to **click on the boxes** to be shown the list of included issues, and then further click on the specific item to get more details. Lastly, for the created vs. resolved charts, the viewer can **mouse hover** to display the number of issues for the specific time and priority.


Relation between issue type and fixing status DEV

Overview More information

Your team is significantly fixing more security debt issues than technical debt issues

Filter time

Last year

Save Cancel

Important issues

Risky issues
1

Unaddressed issues older than a year
5

Total: 6

Filter time

Last month Last 3 months Last 6 months Last year

53 issues found

Average time (in days) required to solve an issue

Total (TD)	High priority (TD)	Critical priority (TD)
187	220	0
Total (SD)	**High priority (SD)**	**Critical priority (SD)**
294	93	0

Average: Leadtime

Technical debt: Created vs. resolved by priority

Last year

Security debt: Created vs. resolved by priority

Last year

Reset scale

Figure 6.2: The entire dashboard

Chapter 7

Discussion

In this chapter I will discuss the results presented in Chapter 4 and the findings from Chapter 5 together with related work. First, I will go through each research question. Then, I will discuss the contributions and implications for research and practice. Lastly, I will explain the quality of this research.

7.1 RQ1 How different is the amount of fixed issues for technical debt and security debt?

There exists several papers looking into the types and amounts of technical debt that is repaid during software evolution. This is shown by the following papers:

While Digkas et al., 2018 analyze the evolution of technical debt in Java projects, investigates Tan et al., 2020 Python projects. In another study, Tan, Feitosa, and Avgeriou, 2022a looks into which types of technical debt that is most likely to be self-fixed (i.e. fixed by the developer who introduces it). Tan et al., 2022b explores the management of technical debt items in both software repositories and issue trackers. They, among other things, look into which types of technical debt items are most likely to be repaid.

This thesis looks into the amount of both technical debt and security debt that is fixed over a given period of time. The earlier identified publications did not take into account the security debt as a type of technical debt. This made it interesting to look into the security perspective of technical debt.

The aim of this research question is to quantify the amount of technical debt and security debt that is repaid and find out which issues are fixed more often. I performed the Chi-Square Test of Independence (section 4.1) which revealed that:

- In general, **security debt** (78% of issues are fixed) is **significantly fixed more** than **technical debt** (61%/62% of issues are fixed)
- For the issues with **low priority**, there was **no significant difference** between the amount of fixed issues for security debt (55% of issues are fixed) and technical debt (45% of issues are fixed)
- For the issues with medium priority, **security debt** (76% of issues are fixed) is **significantly fixed more** than **technical debt** (62%/63% of issues are fixed)
- For the issues with high priority, **security debt** (90% of issues are fixed) is **significantly fixed more** than **technical debt** (72%/71% of issues are fixed)
- For the issues with **critical priority**, there was **no significant difference** between the amount of fixed issues for security debt (94% of issues are fixed) and technical debt (84%/88% of issues are fixed)

Kruke, 2022 found in her study that security debt have a higher priority than technical debt. Moreover, there was a general agreement by her participants that the security issues were fixed first because they could disrupt the sprint (Kruke, 2022). The quantitative findings from this thesis support the qualitative findings in the study of Kruke, 2022.

These results were not surprising for the participant interviewed in the first iteration. It was further elaborated on why the difference for the issues with low priority wasn't big. This is because the teams are encouraged to register most of their debt even though they don't plan to fix it in the nearest future.

The reason for why the security issues are fixed more often than technical debt were not specifically asked for in the interviews in the second iteration (because the main objective were the visualizations) but all of the participants indirectly mentioned why they prioritize security debt over technical debt. One of the mentioned reasons is that by having security debt you have wholes in your security and then it should be fixed to not let hackers in and exploit your system. This is in line with what is said by Kruke, 2022: *"Having security debt makes the system more open to attacks because the system has not reached the security goal due to the security debt"* (p. 98).

The company use different tools to automatically scan their code and look for, among other things, security problems. The sort of tickets that are found by these tools are kept track of in the security index. The index application that the company use were mentioned in relation to the prioritization of debt in all interviews except one. If the issues tracked in the security index are left untouched for a given time, the teams get alerted which might change how they prioritize the issues in the backlog. In addition, these third party tools adds a severity/priority score which tells the software practitioners the risk of the issue. This makes it easier for the teams to prioritize these security issues.

The key difference between technical debt and security debt is the prioritization of the debt (Rindell & Holvitie, 2019). Much effort has been put into estimating the principal and interest of technical debt, but also to set the priorities for repayment schedule (Rindell & Holvitie, 2019). Security debt is usually measured in terms of loss of business if a vulnerability is successfully exploited rather than, for example, source code maintenance effort in traditional technical debt (Martinez et al., 2021). Wrongly prioritizing security items will, in worst case, lead to security incidents and forced payment of the debt with potentially heavy interest. The security incidents also have a multiplicative effect which further complicates estimations. Based on this, the accuracy of risk assessment are crucial for the prioritization of debt items (Rindell & Holvitie, 2019).

In conclusion, security debt has a significant higher fixing rate than technical debt. These results support the literature that exists on security debt. It is also worth mentioning that the index system used by the company gives more penalty points for not addressing security debt than technical debt of the same priority and within the same time frame. Thus, the index system could greatly impact these findings. The penalty point factor will be discussed in section 7.5.

7.2 RQ2 How different is the lead time for technical debt and security debt?

There exists several papers looking into the lead time of technical debt resolution. This is shown by the following papers:

Firstly, Digkas et al., 2018 and Tan et al., 2020 investigates the lead time of different types of technical debt. Secondly, Lenarduzzi, Martini, Saarimäki, and Tamburri, 2021 looks into how the presence of technical debt items in the changed code affects the lead time for resolving the task. Thirdly, Saarimaki, Baldassarre, Lenarduzzi, and Romano, 2019 and Baldassarre, Lenarduzzi, Romano, and Saarimäki, 2020 compares the lead time estimated by SonarQube and the actual time used to fix the issue. Lastly, Tan et al., 2022a is looking into and comparing the lead time of self-fixed and non-self-fixed technical debt.

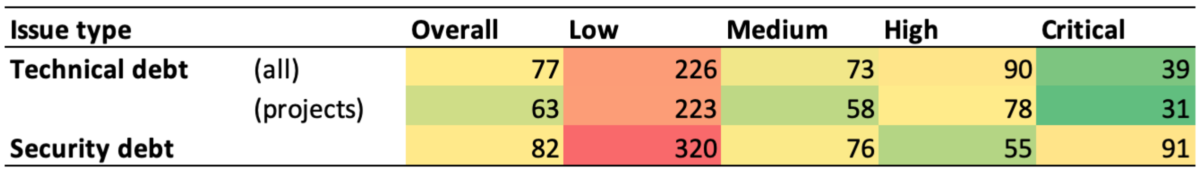
This thesis is looking into the lead time for both technical debt and security debt. As explained in the discussion of the first research question, the same argument is valid seeing as how the security perspective has not been taken into account for the lead time of technical debt resolution.

The aim of this research question is to find out after how long time the issues are repaid and if there is a difference in the lead time for technical debt and security debt. I performed the Mann-Whitney U test (section 4.1) which revealed that:

- In general, the lead time for fixing security debt (Median=82 days) is **higher** than for fixing technical debt (Median=77/63)
- For the issues with low priority, there was no significant difference in the lead time for security debt (Median=320) and technical debt (Median=226/223)
- For the issues with medium priority, the lead time for fixing security debt (Median=76 days) is **higher** than for fixing technical debt (Median=73/58)

- For the issues with high priority, the lead time for fixing security debt (Median=55) is **lower** than for fixing technical debt (Median=90/78)
- For the issues with critical priority, the lead time for fixing security debt (Median=91) is **higher** than for fixing technical debt (Median=39/31)

Table 7.1 shows the median lead time for each issue type by priority. The lower the lead time is the greener is the color, and the higher the lead time, the darker is the shade of red.



Issue type		Overall	Low	Medium	High	Critical
Technical debt	(all)	77	226	73	90	39
	(projects)	63	223	58	78	31
Security debt		82	320	76	55	91

Figure 7.1: Median lead time in days

The results reveal that in general and for all priorities except the high priority, technical debt is fixed in a shorter time than security debt. These findings are inconsistent with the results of the first research question and were surprising to the participant in the first evaluation.

Although all of the participants have said that they use Jira to track their technical debt, there is quite a different practice of how much of the issues the teams register. One of the participants pointed out that if it they could solve the issue in one or two days they didn't register the debt. Some of the teams only register the larger cases and might create one ticket for technical debt that could have been split into ten different items. In contrast, other teams register all debt that they come across and register if they make conscious choices about delaying improvements or refactoring until later. This could impact the findings as some of the issues are not even registered before they are fixed.

When it comes to the findings for the lower priorities, it could be that these issues are considered less important to the teams. It has been said in different interviews that it doesn't matter if the debt with low severity and impact score is open for multiple years because it probably doesn't have much impact on the team. In one of the interviews it was pointed out that they don't calculate and assign issues older than three years because if they haven't done anything for this consistent time it is not required for anyone.

Additionally, it was said in one of the interviews that the size of the project/product could impact the lead time for fixing the debt. This was reasoned by the fact that smaller projects probably have less issues than bigger projects. It was further emphasized that bigger projects might have more cross team collaboration which would make it harder to have an overview of all issues and it would thus require more time to resolve them.

For the issues with high priority, security debt is fixed in shorter time than rest of the technical debt issues which support the participants sayings during the interviews. This was argued to be the case because of the security index. One of the participants also said that if the security debt is both important and haven't been addressed for a long period then something is wrong with the process.

In conclusion, technical debt is fixed in shorter time than security debt for all cases except for the issues with high priority. This was an unexpected result as the previous findings and the participants sayings in the interviews point to the opposite. This might indicate a lack of security knowledge which was explained by Kruke, 2022: *"Having a lack of security knowledge have been described to negatively impact the management activities prevention, identification, evaluation, prioritization, and repayment"* (p.106). The key management activities here are prioritization and repayment due to the focus of this thesis.

7.3 RQ3 How can visualizations of historical data in issue trackers support the prioritizing of the repayment of technical debt and security debt?

There exists some papers looking into visualization techniques in relation to technical debt. I looked into the following papers:

Falessi and Reichel, 2015 and Eliasson, Martini, Kaufmann, and Odeh, 2015 both explore the visualization of the interest/impact of debt, where the first paper looks into technical debt and the second looks into architectural technical debt. When it comes to improving the technical debt management, Power, 2013 reported a visualization technique to help the teams allocate resources for technical debt. The last studied publication, Husby, 2022, developed visualizations of repaid technical debt to improve the technical debt management activities monitoring and communication.

This thesis looks into visualizations not only for technical debt, as explained in the above publications, but also for security debt. How these visualizations can improve the prioritization of the repayment of technical debt and security debt will be explored.

To answer this research question I designed and developed an artefact consisting of four different visualizations with the aim of supporting the planning and prioritization of technical debt and security debt. Below I will compare the intended purpose (requirement) of the visualizations together with the results from the evaluations. Further, I will elaborate on how the visualizations can be used.

Visualization 1: Relation between issue type and resolution

The purpose of this visualization was to give the teams knowledge about if they handle technical debt and security debt equally or not so that they could see if they need to put more focus into the other debt type. The findings from the evaluations proved that this purpose was fulfilled to a certain extent. It was said in most of the interviews that this type of insights were interesting to know about. One of the participants further emphasized this by telling that this visualization could reveal if there is a difference between technical debt and security debt which they haven't really focused on.

However, some of the participants meant that this was a common understanding within the company, i.e., that all teams prioritize security debt over technical debt. In addition, one of the participants found that the created vs. resolved chart revealed more on the difference between how technical debt and security debt were handled.

Visualization 2: Created vs. resolved by priority

The purpose of this visualization was to give the teams knowledge about how many issues they identify and resolve for both technical debt and security debt in a given time. This is because it could reveal if they incur more or less issues over time and if there are periods where they create or fix a lot of issues. The purpose of this visualization was met. One of the participants pointed out that by looking back at these graphs it could reveal discrepancies that should be improved. Moreover, as one of their KPIs is to continuously work on technical debt, this visualization could reveal if they are actually delivering in this area. In addition, most of the participants found it useful to see how the prioritization has been done in the previous period.

Visualization 3: Average lead time

The purpose of this visualization was to give the teams knowledge about how productive they are in order to make it easier for them to take actions if they are slower than expected. This purpose was met. It was said that this visualization could reveal if the lead time is higher than expected. Further, it was mentioned that the visualization could be used to see how long time the team used to solve issues and thus get an idea on how they perform according to their KPIs.

Additionally, the participants pointed out several areas of use for this visualization. One of these was to see if issues should be re-prioritized, i.e., if the issues took longer time than expected to fix it should perhaps have had another priority. Other than this it was said that by setting a goal metric for lead time in relation to technical debt and security debt, this visualization could be useful to look at to yearly evaluate the specified goal.

Visualization 4: Important issues

The purpose of this visualization was to display the issues that are considered important in regard to repayment actions or re-prioritization, so that the teams can more easily act upon it. The purpose of this visualization was was fulfilled to a certain extent. It was said that this way of highlighting important issues could be a useful tool to help prioritize or re-prioritize the issues. Additionally, by looking at this visualization when planning which issues to fix, the highlighted issues can be added as something that should be done during the next period.

However, for one of the teams that didn't have that many issues in their backlog, it was said that this visualization could lead them to being less aware of their issues than they currently are. This was based on how they currently review technical debt and security debt. Because they don't have a lot of tickets and have the time to roughly go through all issues this was preferred over just seeing the most important issues.

7.3.1 Support debt prioritization

The prioritization of technical debt is done by ranking the identified debt according to which items should be repaid first and which items can be allowed until later releases. There have been suggested different approaches to support this activity. The most studied approaches are cost/benefit analysis, high remediation cost first, portfolio approach, and high interest first (Z. Li et al., 2015).

This research introduces a new way of approaching the prioritization of technical debt and security debt. This is done by shedding lights into the differences between security debt and the rest of the technical debt, showing after how much time the debt has been fixed, as well as highlighting the most important issues. The results from the second evaluation describes how the different visualizations can be used to support the planning and prioritization of technical debt and security debt (see section 5.3.2).

The first visualization states whether there is a difference in how the teams prioritize technical debt and security debt. This can be helpful for the teams by putting focus on this difference and thus providing insights into how they have previously handled technical debt and security debt. The second visualization has been proven useful to see how the prioritization within the technical debt and security debt issues have been done in the chosen period. The third visualization reveals how much time the teams have used to fix the issues and could thus reveal if they are using more time than they expect to. This can further support them in finding out if an issue should be re-prioritized. The last visualization displays the most important issues which facilitates which issues should be fixed during the next period.

Some of the teams have a lot of issues in their backlog while others just have a few issues. These differences should be considered when adapting the visualizations. If the list of issues is quite small it is considered manageable and then the visualizations would not give any immediate value in short term. With this I mean that these teams would benefit more from seeing the visualizations over a longer time period in order to discover trends or discrepancies that they would like to improve.

Another thing that should be taking into consideration is that the teams have different goals for their management of the debt. The issues that are considered important to one team might not be the same for another team. Different teams have different KPIs and required levels in the index system. Therefore, the visualizations should reflect these kinds of issues. For example, the teams should define which issues are considered important and these will then be visualized as the most important issues.

All of the participants have pointed out that the visualizations could add value in their work on technical debt and security debt. I experienced that the participants were genuinely interested in this research and that the artefact inspired them to think differently about how they manage technical debt and security debt. Several of the participants have said that they would really like to have something similar to the presented visualizations in their practice.

To conclude, the artefact can be considered a starting point of an approach to debt prioritization. It is crucial that the visualizations reflects and considers the amount of technical debt and security debt found in their system.

7.4 Contributions

The following bullet points summarize the contributions of this thesis:

- Initial evidence have been found to show that security debt has a higher significant fixing rate than technical debt.
- The calculation of the lead time has been presented and shows that technical debt often is fixed faster than security debt.
- Earlier publications does not have a substantial focus on visualizations related to technical debt or security debt. This study contributes to this area by producing a dashboard containing four distinct technical debt and security debt visualizations.
- An artefact has been provided as a starting point for how visualizations of historical data in issue trackers can support the planning and prioritization of technical debt and security debt. The artefact has been initially validated with a limited number of teams.

7.4.1 Implications for research

- All results in this thesis are impacted by the penalizing strategies used within the company. Further research should compare the results from this thesis to other company contexts.
- The results from RQ1 shows that security debt has a significantly higher fixing rate than technical debt. This might be due to the index system used in the company. Further research should focus on the amount of issues that are fixed for technical debt and security debt in a context where they don't use penalizing strategies.
- The results from RQ2 shows that technical debt in general is fixed in a shorter time period than security debt. In this research I was not able to consider issues size due to data set restrictions. Further research should explore the additional factor of issue size.
- The results from RQ2 shows that technical debt in general is fixed in a shorter time period than security debt. This might be due to lack of security knowledge. Further research should investigate this.

- I presented a starting point for visualizing historical data in issue trackers (e.g. Jira) for the purpose of supporting technical debt and security debt planning and prioritization. Further research should look into improvements on these visualizations.
- The general consensus after talking to the participants showed that the visualizations made them rethink their management of technical debt and security debt. As they all pointed out that the use of visualizations in relation to technical debt can be valuable, further research should investigate how visualizations can support the management of technical debt.

7.4.2 Implications for practice

- From RQ1 it can be interpreted that software practitioners should have an understanding of their fixing ratio of their technical debt and security debt. Based on these ratios they might be able to better plan and prioritize the debt.
- From RQ2 it can be found that lead time can be helpful for the teams in relation to debt management. This metric can be used to show the team's productivity of debt repayment.
- From RQ3 visualizations have been presented. Teams within the company can take inspiration from the presented artefact and choose the visualizations that could give them value in terms of supporting the prioritization and planning process.
- From RQ3 visualizations have been presented. Teams outside the company might get similar results in terms of re-thinking their debt management strategies due to the visualizations showing an overview of how they previously have prioritized and repaid the debt.

7.5 Validity and limitations

In this section I will first elaborate on the threats to validity. Then, I will point out the limitations in this study. Lastly, I will describe how I have followed the seven Design Science Research guidelines provided by Hevner et al., 2004 during my research.

7.5.1 Validity

"The validity of a study denotes the trustworthiness of the results, to what extent the results are true and not biased by the researchers' subjective point of view" (Runeson & Höst, 2009, p. 153). There are four different ways to classify threats to validity in the literature: construct validity, internal validity, external validity, and reliability. Lastly, I will discuss conclusion validity as it is important to ensure the quality of the quantitative data analysis (García-Pérez, 2012).

Construct validity

Construct validity reflects the extent to which the operational measures being studied really represent what the researcher has in mind and what is being investigated according to the research questions (Runeson & Höst, 2009).

As a common threat to construct validity is that the constructs discussed in the interview questions are not interpreted in the same way by the researcher and the interviewees (Runeson & Höst, 2009) it was important to me that we had a common understanding of the terms technical debt and security debt. Because of this, the interviews started by discussing the concept of technical debt. In addition, I explained how I classified the security debt items.

A threat to the construct validity was when I classified the security issues in the data set. The classification were done by searching for specific *security* words according to the taxonomy proposed by Firesmith, 2003. This can result in false negatives, i.e., the issue can be a security issue although it is not tagged with one of these security words.

Internal validity

Internal validity is of concerns when causal relations are examined (Runeson & Höst, 2009).

A threat to the internal validity is that the size of an issue can impact the lead time for fixing the issue. As I didn't have information about the issue size, I have not considered the size when looking into the difference in lead time for fixing technical debt and security debt.

Another threat to the internal validity is that I spoke to different teams and each team use different ways of prioritizing the debt.

External validity

External validity concerns the extent to which it is possible to generalize the findings, and the extent to which the findings are of interest to other people outside the investigated case (Runeson & Höst, 2009).

The analyzed data set contains data from different projects within one company. This is limiting the ability to generalize the conclusions regarding lead time and amount of fixed technical debt and security debt to projects in a different company.

The studied company uses Jira to track their issues. The artefact is created in Jira and requires the intended user to track their issues. This is limiting the ability to generalize the conclusions related to the artefact to companies that does not use Jira to track their technical debt.

Reliability

Reliability is concerned with to what extent the data and the analysis are dependent on the specific researchers (Runeson & Höst, 2009).

A common threat to the reliability in relation to qualitative research is that the researcher influences the respondents and the overall process. In order to mitigate my own bias I had discussions with the company stakeholders and my supervisors continuously throughout the thesis. This was important because I was the only person who analyzed and interpreted the data. Additionally, to mitigate the threat of my own bias throughout the thesis I did triangulation of the sources of data: document analysis and interviews.

Conclusion validity

Conclusion validity concerns the use of appropriate statistical methods which behaves accurately. Additionally, the statistical method must provide an answer to the research question (García-Pérez, 2012).

I have reported results considering inferential statistics and used two-population hypothesis testing. To answer the first research question I performed the Chi-Square Test of Independence which requires that the expected count for each issue-fixing combination is at least 5. This is not the case for the security issues of critical priority. Because of this, the results for RQ1 regarding the critical issues are not as reliable as the other results.

For RQ2 I have performed the non-parametric test, Mann-Whitney U test, according to the normality of the data. I used Xlstat to perform all statistical analyses since it gives good confidence regarding the results quality. In addition, I performed the tests several times to see that I obtained the same results.

Limitations

Few participants
For the evaluation of the artefact I conducted only six interviews. The fact that the evaluation is based on few participants is a limitation in this study.

Issue size
I have not considered the size of the issue when calculating the lead time for fixing technical debt and security debt because I didn't have information about this.

Time constraint

For the first five interviews the interviews lasted approximately 1 hour. But for the last interview I only had 30 minutes which limited the questions and feedback I could get.

Impact of penalty points
The index system used by the company might impact how the debt is prioritized. This has also been mentioned by several participants. Due to time constraint I could not look into this.

Design Science Research Guidelines

Guideline 1: Design as an Artefact
This first guideline implies that the research must produce a usable artefact in the form of a construct, model, method, or an instantiation (Hevner et al., 2004). The artefact provided in my thesis can be considered an instantiation. It is a dashboard of history keeping (implemented in Jira) which aims to support in the planning and prioritization of technical debt and security debt. Chapter 5 describes how the artefact has been designed and developed through two iterations. The final artefact is presented in Chapter 6.

Guideline 2: Problem relevance
This guideline implies that the objective of the research is to create technological solutions to important and relevant problems (Hevner et al., 2004). The artefact provided in this research address a relevant and important problem. Before I started to design and develop the artefact, I got input from relevant stakeholders within the company. I also looked into the current literature on technical debt and security debt. Further, I used document analysis where I got knowledge of the data available. Then, by performing statistical tests I got to know the environment/context relevant to the problem. Based on the knowledge I got from the document analysis and statistical tests, I discussed different ideas together with the company stakeholders. This resulted in the requirements proposed to solve the problem. These requirements were used as a basis for the development of the artefact.

Guideline 3: Design evaluation
This guideline implies that the utility, quality, and efficacy of the artefact must

be rigorously demonstrated using well-conducted evaluation methods (Hevner et al., 2004). The artefact was evaluated by conducting interviews with relevant stakeholders within two iterations. This is described in Chapter 3. The understandability and usefulness of the artefact is presented in section 5.3.2.

Guideline 4: Research Contributions
This guideline implies that the research must provide clear and verifiable contributions in the areas of the designed artefact (Hevner et al., 2004). The contributions in the area of the designed artefact is pointed out later in this chapter (see section 7.4).

Guideline 5: Research Rigor
This guideline implies that the research is dependent on the use of rigorous methods in both the construction and evaluation of the designed artefact (Hevner et al., 2004). The artefact is designed and developed based on the company stakeholders proposed requirements. Additionally, I have gotten continuous feedback and input from these stakeholders during this project.

Guideline 6: Design as a search process
This guideline implies that the search for an effective artefact requires the use of available means to reach desired goals while satisfying laws of the problem environment (Hevner et al., 2004). The artefact has evolved through two iterations. The progress has been made iterative as the scope of the problem expands.

Guideline 7: Communication of research
This last guideline implies that the research must be presented effectively both to technology-oriented as well as management-oriented audiences (Hevner et al., 2004). The artefact has been evaluated by both technology-oriented and management-oriented audiences. Further, both perspectives are presented in the evaluation (see section 5.3.2).

Chapter 8

Conclusion

In this thesis, I have focused on the difference between technical debt and security debt. For the first research question I have looked into how different the amount of fixed issues are. For the second research question I have explored the difference in the lead time for fixing an issue. Finally, for the last research question I have investigated how visualizations of historical data in Jira can support teams in prioritizing the repayment of technical debt and security debt.

To answer the research questions I have conducted a Design Science Research study. A technical debt data set from the company was analyzed to find out which issues have been repaid the most and which issues have been repaid in the shortest time. The findings from the quantitative data analysis together with requirements proposed by company stakeholders resulted in an artefact containing four distinct visualizations of historical technical debt and security debt issues. The artefact have been evaluated through six interviews within two iterations.

The results from the first research question showed that security debt had a significant higher fixing rate than technical debt. Regarding the second question, the results revealed that technical debt were often fixed in a shorter time than security debt. Finally, the findings from the last research question initially proved that the created artefact could support teams in prioritizing the repayment of technical debt and security debt.

Historical data from Jira can provide valuable information about how technical debt and security debt have been prioritized in the past. The two first research questions provided both context findings for the rest of the thesis as well as interesting empirical findings for research purposes. For the last research question, the artefact offered a new way of approaching the prioritization of debt by providing insights into the differences between security debt and the rest of technical debt, showing after how much time the debt have been fixed, as well as highlighting the most important unaddressed issues.

Bibliography

Ahmadjee, S., & Bahsoon, R. (2019). *A taxonomy for understanding the security technical debts in blockchain based systems*.

Alfayez, R., Alwehaibi, W., Winn, R., Venson, E., & Boehm, B. (2020). A systematic literature review of technical debt prioritization. In *Proceedings of the 3rd international conference on technical debt* (pp. 1–10). doi:10.1145/3387906.3388630

Alves, N. S., Mendes, T. S., de Mendonça, M. G., Spínola, R. O., Shull, F., & Seaman, C. (2016). Identification and management of technical debt: A systematic mapping study. *Information and Software Technology*, *70*, 100–121. doi:https://doi.org/10.1016/j.infsof.2015.10.008

Atlassian. (2022a). Retrieved from https://developer.atlassian.com/cloud/jira/platform/index/

Atlassian. (2022b). *Dashboard gadget*. Retrieved from https://developer.atlassian.com/platform/forge/ui-kit-components/jira/dashboard-gadget/

Atlassian. (2022c). *What is jira used for?* Retrieved from https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for#Jira-for-requirements-&-test-case-management

Baldassarre, M. T., Lenarduzzi, V., Romano, S., & Saarimäki, N. (2020). On the diffuseness of technical debt items and accuracy of remediation time when using sonarqube. *Information and Software Technology*, *128*, 106377. doi:https://doi.org/10.1016/j.infsof.2020.106377

Beatty, W. (2018). Decision support using nonparametric statistics. Cham: Springer International Publishing : Imprint: Springer.

Bowen, G. (2009). Document analysis as a qualitative research method. *Qualitative Research Journal*, *9*, 27–40. doi:10.3316/QRJ0902027

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*, 77–101. doi:10.1191/1478088706qp063oa

Brocke, J. v., Hevner, A., & Maedche, A. (2020). Introduction to design science research. (pp. 1–13). doi:10.1007/978-3-030-46781-4_1

Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., . . . Zazworka, N. (2010). Managing technical debt in software-reliant systems. doi:10.1145/1882362.1882373

Ciolkowski, M., Lenarduzzi, V., & Martini, A. (2021). 10 years of technical debt research and practice: Past, present, and future. *IEEE Software*, *38*(6), 24–29. doi:10.1109/MS.2021.3105625

Codabux, Z., & Williams, B. (2013). Managing technical debt: An industrial case study. In *Proceedings of the 4th international workshop on managing technical debt* (pp. 8–15). San Francisco, California: IEEE Press.

Digkas, G., Lungu, M., Avgeriou, P., Chatzigeorgiou, A., & Ampatzoglou, A. (2018). How do developers fix issues and pay back technical debt in the apache ecosystem? In *2018 ieee 25th international conference on software analysis, evolution and reengineering (saner)* (pp. 153–163). doi:10.1109/SANER.2018.8330205

Digkas, G., Lungu, M., Chatzigeorgiou, A., & Avgeriou, P. (2017). The evolution of technical debt in the apache ecosystem. In A. Lopes & R. de Lemos (Eds.), *Software architecture* (pp. 51–66). Cham: Springer International Publishing.

Eliasson, U., Martini, A., Kaufmann, R., & Odeh, S. (2015). Identifying and visualizing architectural debt and its efficiency interest in the automotive domain: A case study. In *2015 ieee 7th international workshop on managing technical debt (mtd)* (pp. 33–40). doi:10.1109/MTD.2015.7332622

Falessi, D., & Reichel, A. (2015). Towards an open-source tool for measuring and visualizing the interest of technical debt. In *2015 ieee 7th international workshop on managing technical debt (mtd)* (pp. 1–8). doi:10.1109/MTD.2015.7332618

Firesmith, D. (2003). *Common concepts underlying safety, security, and survivability engineering* (tech. rep. No. CMU/SEI-2003-TN-033). Software Engineering Institute, Carnegie Mellon University. Pittsburgh, PA. Retrieved from http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6553

Fowler, M. (2009). Technical debt quadrant. Retrieved from https://martinfowler.com/bliki/TechnicalDebtQuadrant.html

García-Pérez, M. (2012). Statistical conclusion validity: Some common threats and simple remedies. *Frontiers in Psychology*, *3*. doi:10.3389/fpsyg.2012.00325

Guo, Y., Seaman, C., Gomes, R., Cavalcanti, A., Tonin, G., Da Silva, F. Q. B., ... Siebra, C. (2011). Tracking technical debt — an exploratory case study. In *2011 27th ieee international conference on software maintenance (icsm)* (pp. 528–531). doi:10.1109/ICSM.2011.6080824

Hevner, A., R, A., March, S., T, S., Park, Park, J., ... Sudha. (2004). Design science in information systems research. *Management Information Systems Quarterly*, *28*, 75–.

Husby, V. R. (2022). *Visualizing historical project data to improve technical debt management* (Master's thesis).

Johannesson, P. (2014). An introduction to design science. Cham: Springer International Publishing : Imprint: Springer.

Johannesson, P., & Perjons, E. (2014). *An introduction to design science*. doi:https://doi.org/10.1007/978-3-319-10632-8

Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. *IEEE Software*, *29*. doi:10.1109/MS.2012.167

Kruchten, P., Nord, R. L., Ozkaya, I., & Falessi, D. (2013). Technical debt: Towards a crisper definition report on the 4th international workshop on managing technical debt. *38*(5). doi:10.1145/2507288.2507326

Kruke, M. M. (2022). *Security debt in practice* (Master's thesis).

Lanza, M. (2001). The evolution matrix: Recovering software evolution using software visualization techniques. In *Proceedings of the 4th international workshop on principles of software evolution* (pp. 37–42). doi:10.1145/602461.602467

Lenarduzzi, V., Martini, A., Saarimäki, N., & Tamburri, D. A. (2021). Technical debt impacting lead-times: An exploratory study. In *2021 47th euromicro conference on software engineering and advanced applications (seaa)* (pp. 188–195). doi:10.1109/SEAA53835.2021.00032

Li, Y., Soliman, M., & Avgeriou, P. (2020). Identification and remediation of self-admitted technical debt in issue trackers. In *2020 46th euromicro conference on software engineering and advanced applications (seaa)* (pp. 495–503). doi:10.1109/SEAA51224.2020.00083

Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. *The Journal of systems and software*, *101*, 193–220.

Lindgren, M., Wall, A., Land, R., & Norström, C. (2008). A method for balancing short- and long-term investments: Quality vs. features. In *2008 34th euromicro conference software engineering and advanced applications* (pp. 175–182). doi:10.1109/SEAA.2008.22

Martinez, J., Quintano, N., Ruiz, A., Santamaria, I., de Soria, I. M., & Arias, J. (2021). Security debt: Characteristics, product life-cycle integration and items. In *2021 ieee/acm international conference on technical debt (techdebt)* (pp. 1–5). doi:10.1109/TechDebt52882.2021.00009

Martini, A., Besker, T., & Bosch, J. (2018). Technical debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations. *Science of Computer Programming*, *163*, 42–61. doi:https://doi.org/10.1016/j.scico.2018.03.007

Martini, A., Bosch, J., & Chaudron, M. (2015). Investigating architectural technical debt accumulation and refactoring over time: A multiple-case study. *Information and Software Technology*, *67*, 237–253. doi:https://doi.org/10.1016/j.infsof.2015.07.005

Mendes, T., Gomes, F., Gonçalves, D., & et al. (2019). Visminertd: A tool for automatic identification and interactive monitoring of the evolution of technical debt items. *J Braz Comput Soc*, *25*. doi:https://doi.org/10.1186/s13173-018-0083-1

Mostowski, P., Kuder, A., Filipczak, J., & Rutkowski, P. (2018). Workflow management and quality control in the development of the PJM corpus: The use of an issue-tracking system. In M. Bono, E. Efthimiou, S.-E. Fotinea, T. Hanke, J. A. Hochgesang, J. Kristoffersen, ... Y. Osugi (Eds.), *11th international conference on language resources and evaluation (LREC 2018): Proceedings of the LREC2018 8th workshop on the representation and processing of sign languages: Involving the language community* (pp. 133–138). Miyazaki, Japan: European Language Resources Association (ELRA). Retrieved from https://www.sign-lang.uni-hamburg.de/lrec/pub/18045.pdf

Novais, R. L., Torres, A., Mendes, T. S., Mendonça, M., & Zazworka, N. (2013). Software evolution visualization: A systematic mapping study. *Information* *and Software Technology*, *55*(11), 1860–1883. doi:https://doi.org/10.1016/j.infsof.2013.05.008

Ortu, M., Destefanis, G., Kassab, M., & Marchesi, M. (2015). Measuring and understanding the effectiveness of jira developers communities. doi:10.1109/WETSoM.2015.10

Ortu, M., Destefanis, G., Murgia, A., Tonelli, R., Marchesi, M., & Adams, B. (2015). The jira repository dataset: Understanding social aspects of software development.

Ortu, M., Murgia, A., Destefanis, G., Tourani, P., Tonelli, R., Marchesi, M., & Adams, B. (2016). The emotional side of software developers in jira. In *2016 ieee/acm 13th working conference on mining software repositories (msr)* (pp. 480–483).

Paredes, J., Anslow, C., & Maurer, F. (2014). Information visualization for agile software development. In *2014 second ieee working conference on software visualization* (pp. 157–166). doi:10.1109/VISSOFT.2014.32

Peffers, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*, 45–77.

Pinzger, M., Gall, H., Fischer, M., & Lanza, M. (2005). Visualizing multiple evolution metrics. In *Proceedings of the 2005 acm symposium on software visualization* (pp. 67–75). doi:10.1145/1056018.1056027

Power, K. (2013). Understanding the impact of technical debt on the capacity and velocity of teams and organizations: Viewing team and organization capacity as a portfolio of real options. In *2013 4th international workshop on managing technical debt (mtd)* (pp. 28–31). doi:10.1109/MTD.2013.6608675

Rindell, K., Bernsmed, K., & Jaatun, M. (2019). Managing security in software or: How i learned to stop worrying and manage the security technical debt. doi:10.1145/3339252.3340338

Rindell, K., & Holvitie, J. (2019). Security risk assessment and management as technical debt. doi:10.1109/CyberSecPODS.2019.8885100

Rios, N., de Mendonça Neto, M. G., & Spínola, R. O. (2018). A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. *Information and Software Technology*, *102*, 117–145. doi:https://doi.org/10.1016/j.infsof.2018.05.010

Rumsey, D. (2009). Statistics ii for dummies. Hoboken, N.J: Wiley.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *14*(2). doi:10.1007/s10664-008-9102-8

Saarimaki, N., Baldassarre, M. T., Lenarduzzi, V., & Romano, S. (2019). On the accuracy of sonarqube technical debt remediation time. In *2019 45th euromicro conference on software engineering and advanced applications (seaa)* (pp. 317–324). doi:10.1109/SEAA.2019.00055

Serrano, N., & Ciordia, I. (2005). Bugzilla, itracker, and other bug trackers. *IEEE Software*, *22*(2), 11–13. doi:10.1109/MS.2005.32

Siavvas, M., Tsoukalas, D., Jankovic, M., Kehagias, D., Chatzigeorgiou, A., Tzovaras, D., . . . Gelenbe, E. (2019). *An empirical evaluation of the relationship between technical debt and software security*. doi:10.13140/RG.2.2.15488.79365

Siavvas, M., Tsoukalas, D., Jankovic, M., Kehagias, D., & Tzovaras, D. (2020). Technical debt as an indicator of software security risk: A machine learning approach for software development enterprises. *Enterprise Information Systems*, *0*(0), 1–43. doi:10.1080/17517575.2020.1824017

Sneed, H. M. (2014). Dealing with technical debt in agile development projects. In D. Winkler, S. Biffl, & J. Bergsmann (Eds.), *Software quality. model-based approaches for advanced software and systems engineering* (pp. 48–62). Cham: Springer International Publishing.

Tan, J., Feitosa, D., & Avgeriou, P. (2022a). Does it matter who pays back technical debt? an empirical study of self-fixed td. *Information and Software Technology*, *143*, 106738. doi:https://doi.org/10.1016/j.infsof.2021.106738

Tan, J., Feitosa, D., & Avgeriou, P. (2022b). The lifecycle of technical debt that manifest in both source code and issue trackers. *SSRN Electronic Journal*. doi:10.2139/ssrn.4160012

Tan, J., Feitosa, D., Avgeriou, P., & Lungu, M. (2020). Evolution of technical debt remediation in python: A case study on the apache software ecosystem. *Software: Evolution and Process*. doi:https://doi.org/10.1002/smr.2319

Vathsavayi, S. H., & Systä, K. (2016). Technical debt management with genetic algorithms. In *2016 42th euromicro conference on software engineering and advanced applications (seaa)* (pp. 50–53). doi:10.1109/SEAA.2016.43

Vinutha, H. P., Poornima, B., & Sagar, B. M. (2018). Detection of outliers using interquartile range technique from intrusion dataset. In S. C. Satapathy,

J. M. R. Tavares, V. Bhateja, & J. R. Mohanty (Eds.), *Information and decision sciences* (pp. 511–518). Singapore: Springer Singapore.

Yli-Huumo, J., Maglyas, A., & Smolander, K. (2016). How do software development teams manage technical debt? – an empirical study. *Journal of Systems and Software*, *120*. doi:10.1016/j.jss.2016.05.018

A Interview guide (first iteration)

Interview guide (first iteration)

Consent
Thanks for participating Introduction of study Documenting consent
Get to know participant
- Please tell me about yourself and what you do at [company] - Role?
Statistical findings
Show results from both statistical tests - What did you expect? - Why do you think the results are like they are?
Visualization 1 (Chi-Square)
Explain visualization and purpose - What do you think of this? - What kind of information would you like? - Useful? For whom? When? - Improvements?
Visualization 2 (Created vs. resolved)
Explain visualization and purpose - What do you think of this? - Useful? For whom? When? - Improvements?
Visualization 3 (Avg. lead time)
Explain visualization and purpose - What do you think of this? - Useful? For whom? When? - Improvements?
Visualization 4 (Lead time chart)
Explain visualization and purpose - What do you think of this? - Useful? For whom? When? - Improvements?
Visualization 5/6 (Lists)
Explain visualization and purpose - What do you think of this? - Useful? For whom? When? - Improvements?
Closing interview (5 min)
- Any ideas on other visualizations that could help the teams in the planning and prioritization? - What haven't I asked you today that you think would be valuable for me to know? - May I contact you if I have any other questions on this topic?

B Interview guide (second iteration)

Interview guide (second iteration)

Consent (5 min)

Thanks for participating
Introduction of study
Documenting consent

Opening interview (20 min)

Get to know participant

- Please tell me about yourself and what you do at [company]
- Role?
- Background?
- Team size?

User behavior

- What solutions do you use to track issues?
- How is you teams «practice» of registering TD/SD in Jira? (Registering all issues or only larges issues?)
  - labels?
- How many open TD and SD issues in Jira?
- How do you prioritize TD and SD?
- After registering TD/SD in Jira, how do you decide which items to fix and not fix?
  - how much time do you spend on planning?
  - how often do you plan?
- Currently using any visualizations in Jira (or other places) as a part of the planning and prioritization of TD/SD?
  - What kinds of visualizations?
  - What do you like about these visualizations?
  - What do you not like about these visualizations?
- Can you think of any visualizations that could make this planning/prioritizing process easier?

Solution process

- Under what circumstances would you want to receive an alert in Jira? (Maximum amount of time that a task could be in Jira?)

*** Share dashboard *** (30 min)

- What do you see here? What are you looking at now?

Visualization 1

Demonstrate visualization

- What do you think this visualization is for?
- How could it be used? What could it be used for?
- Who would have used it?
- When would it be used?
- What do you think is missing?

Visualization 2

Demonstrate visualization

- What do you think this visualization is for?
- How could it be used? What could it be used for?
- Who would have used it?
- When would it be used?
- What do you think is missing?

Visualization 3

Demonstrate visualization - What do you think this visualization is for? - How could it be used? What could it be used for? - Who would have used it? - When would it be used? - What do you think is missing?
Visualization 4
Demonstrate visualization - What do you think this visualization is for? - How could it be used? What could it be used for? - Who would have used it? - When would it be used? - What do you think is missing?
Closing interview (5 min)
- What haven’t I asked you today that you think would be valuable for me to know? - May I contact you if I have any other questions on this topic?

C Observed and expected counts

Priority	Issue type	Fixed	Not fixed
Overall	TD (all)	5007	3152
		5176	**2983**
	TD (projects)	4573	2767
		4730	**2610**
	SD	929	269
		760	**438**
Low	TD (all)	238	290
		247	**281**
	TD (projects)	223	269
		232	**260**
	SD	59	48
		50	**57**
Medium	TD (all)	4255	2655
		4364	**2546**
	TD (projects)	3899	2335
		4000	**2234**
	SD	630	195
		521	**304**
High	TD (all)	396	157
		427	**126**
	TD (projects)	363	151
		395	**119**
	SD	223	25
		192	**56**
Critical	TD (all)	118	23
		120	**21**
	TD (projects)	88	12
		89	**11**
	SD	17	1
		15	3

Table 8.1: Observed and expected counts for the Chi-Square Test of Independence. The expected counts are shown in bold and rounded to its nearest integer.