Master's thesis

Kristoffer Håkon Håkonsen

Triggering threat modeling in agile development

A new method artifact

Master's thesis in Computer Science
Supervisor: Daniela Soares Cruzes
June 2022

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Kristoffer Håkon Håkonsen

Triggering threat modeling in agile development

A new method artifact

Master’s thesis in Computer Science
Supervisor: Daniela Soares Cruzes
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

In agile development, threat modeling should be performed as early as possible, but it will need to be updated as the architecture and system change. The challenge for an agile development team then becomes to determine when to update the threat modeling. The current body of research lacks a systematic way of determining when this update should be triggered based on the changes made to the system.
This thesis aims to improve software security in agile development by proposing a method-artifact for triggering threat modeling. More specifically, it answers the research questions:
RQ1: How can threat modeling be triggered based on code changes?
RQ2: What is the effect of the suggested approach from RQ1?

The method-artifact proposed in this thesis categorizes code changes and creates triggers for threat modeling based on the actual changes to the system in a systematic manner. The artifact was evaluated with two paper prototype user tests at the IT company SpareBank 1 Utvikling to improve and evaluate the artifact. Later, a minimum-viable-product implementation was tested at the software development company Visma as part of their pipeline in GitHub.

This thesis follows the design science methodology, and the structure is, therefore, a combination of aIMRAD and the steps in design science. It begins by explicating the problem, outlining an artifact, designing and developing the artifact, followed by a demonstration of the artifact on a fictional case. Finally, the artifact is evaluated against how well it solves the problem and fulfills the requirements, its side effects are studied, the usability is evaluated, and it is compared to other similar artifacts.

The results indicate that the artifact effectively solved the problems and security experts rated it highly. The usability was within the acceptable range, and most usability challenges could be improved by increased automation or minor tweaks. The artifact impacted the developer's agile workflow to a small extent and required little extra time and effort.

Sammendrag

Trussemodellering burde gjennomføres så tidlig som mulig i smidig utvikling, men den må oppdateres etter hvert som arkitekturen og systemet endres. I smidig utvikling er det utfordrende å vite når denne oppdateringen bør gjennomføres. Den gjeldende akademiske litteraturen mangler en systematisk måte å utløse slike oppdateringer basert på endringer gjort i koden. Denne masteroppgaven har som mål å forbedre programvaresikkerheten i smidig utvikling ved å foreslå et metode-artefakt som kan indikere når trussemodellering bør finne sted. Mer spesifikt prøver denne oppgaven å besvare følgende forskningsspørsmål:
Forskningsspørsmål 1: Hvordan kan trussemodellering bli utløst basert på kodeendringer?
Forskningsspørsmål 2: Hva er effekten av metoden som løser forskningsspørsmål 1?

Metode-artefaktet som er foreslått i denne oppgaven kategoriserer kodeendringer og utløser trussemodelleringer basert på de faktiske endringene i systemet, på en systematisk måte. Artefaktet ble forbedret og evaluert med to papir prototype brukertester hos IT selskapet SpareBank 1 Utvikling. Senere ble en minimal implementasjon av artefaktet testet hos IT utviklingsselskapet Visma, som en del av deres arbeidsflyt i GitHub.

Denne oppgaven følger design forsknings metodologien, og strukturen i oppgaven er derfor en blanding av AIMRAD og stegene i design forskning. Den begynner med å utforske og forklare problemet, lage et utkast av artefaktet, etterfulgt av design og implementasjon av artefaktet, deretter en demonstrasjon i en fiktiv case. Til slutt ble artefaktet evaluert etter hvor bra det løser problemet, hvor godt den fyller kravene som er definert, mulige bi-effekter ble undersøkt, brukbarheten ble evaluert, og den ble sammenlignet med andre artefakter.

Resultatene indikerer at artefaktet er effektivt for å løse problemet og sikkerhetseksperter rangerer den høyt. Brukbarheten var innenfor de akseptable verdier, og de fleste brukbarhetsutfordringene kan forbedres med automatisering eller små forbedringer. Artefaktet påvirket utviklernes smidige arbeidsflyt lite, og den krevde lite ekstra tid og innsats.

Acknowledgment

This master thesis was written at the Department of Computer Science at NTNU in the period January 2022 to June 2022 as part of the course TDT4900 - Computer Science, Master's Thesis.

I will begin by thanking my supervisor, Professor Daniela Soares Cruzes, at NTNU, who has provided guidance and suggestions throughout this semester, as well as providing me with contacts in the industry. This thesis would not have been possible without her continued support.

This thesis has been created with the collaboration of SpareBank 1 Utvikling and Visma. I would like to thank the Head of Security Development at Visma, Dr. Monica Iovan, who has provided feedback and confidence in the artifact and ensured that the artifact could be tested with a feature team at Visma.
The two senior infrastructure engineers, Nicoleta Botosan and Romina Druta at Visma, also deserve great thanks for their assistance and feedback during the implementation of this artifact before the case study. In addition, I would like to thank the feature team at Visma that tested this artifact.
I would like to thank the Software development leaders Marthe Slaatsveen and Lars Kirkhus at SpareBank 1 Utvikling for making the collaboration possible. The two security architects from SpareBank 1 Utvikling, Jon Are Rakvåg and Adrian Alexander Eriksen, deserve great thanks for their collaboration and support throughout this thesis. I would also thank the two teams at SpareBank 1 Utvikling that tested this artifact.

I would like to extend a special thanks to the external security experts who provided me with feedback, recommendations, suggestions, and confidence throughout this thesis. These experts include but are not limited to Matt Konda and Rohini Sulatycki.

My fellow master's student has been instrumental in the creation of this thesis by providing me with a positive and humorous environment throughout the work.

I would also like to thank my mother and father for assisting with technical and emotional support throughout this thesis.

Finally, I would like to thank my fiance, Sara, for her constant love and support during the work on this thesis. I am grateful that you survived and listened to my endless explanations, and provided me with regular sanity checking of my various ideas.

Contents

Abstract . . . iii
Sammendrag . . . v
Acknowledgment . . . vii
Contents . . . ix
Figures . . . xiii
Tables . . . xv
1 Introduction . . . 1
2 Background . . . 3
2.1 Threat Modeling . . . 3
2.2 Pre-study on Threat modeling . . . 3
2.3 Threat modeling in agile development . . . 4
2.4 Triggering threat modeling . . . 5
3 Method . . . 9
3.1 Research Methodology . . . 9
3.1.1 Design Science . . . 9
3.1.2 A method framework for Design Science Research . . . 9
3.2 Explicate Problem . . . 11
3.2.1 Define Requirements . . . 14
3.3 Data collection methods . . . 17
3.3.1 User testing . . . 17
3.3.2 Case study . . . 20
3.3.3 Expert discussions . . . 22
3.4 Data Analysis . . . 22
3.5 Ethics . . . 23
4 Triggering threat modeling . . . 25
4.1 Design and Develop Artifact . . . 25
4.1.1 The design of the artifact . . . 25
4.1.2 Artifact description . . . 28
4.2 Demonstrate Artifact . . . 29
4.2.1 Theoretical demonstration . . . 30
4.3 Case study implementation . . . 31
4.4 Case study raw data . . . 35
5 Evaluation . . . 41
5.1 Data generation . . . 41

5.1.1 User tests . . . 41
5.1.2 Case study at Visma with GitHub action . . . 42
5.2 Evaluation descriptions . . . 42
5.2.1 To what extent is the artifact effective for solving the problem? . . . 43
5.2.2 Usability . . . 43
5.2.3 Evaluate requirements . . . 43
5.2.4 Compare artifacts to similar artifacts . . . 44
5.2.5 Investigate side-effects . . . 44
5.2.6 Formative evaluation . . . 44
5.3 Evaluation results . . . 44
5.3.1 Participants demographics . . . 44
5.3.2 To what extent is the artifact effective for solving the problem? . . . 45
5.3.3 Usability testing . . . 47
5.3.4 Evaluate requirements . . . 50
5.3.5 Compare artifacts to other . . . 53
5.3.6 Investigate side-effects . . . 54
5.3.7 Formative evaluation . . . 55
6 Discussion . . . 57
6.1 Research questions . . . 57
6.2 Implications for research . . . 60
6.3 Implications to practice . . . 60
6.4 Limitations and threats to validity . . . 61
6.5 Reflections . . . 62
7 Conclusion . . . 63
7.1 Conclusion . . . 63
7.2 Future work . . . 64
Bibliography . . . 65
A Artifact demonstration . . . 69
A.1 The demonstration categories . . . 69
A.2 The demonstration pull requests . . . 70
A.3 Demonstration calculations . . . 71
B NSD notification forms . . . 73
B.1 User tests . . . 73
B.2 Case study . . . 77
C Evaluation details . . . 81
C.1 Paper prototype user tests . . . 81
C.1.1 The paper prototype . . . 81
C.1.2 Interview guide . . . 81
C.2 The theoretical description . . . 88
C.2.1 Results . . . 88
C.3 Case study . . . 97
C.3.1 Interview guide . . . 97
C.4 The theoretical description . . . 104
C.5 Results . . . 104

C.5.1 Interview with participants 104
C.5.2 Interview with the head of security development and Senior Infrastructure Engineer 109
C.6 SUS form 109
C.7 Linear transformation scales 110
D Implemented artifact output 113
D.1 The questions 113
D.2 Raw output 114
D.3 The accumulated output without contribution factors 118
D.4 The contribution factors 122
D.5 The accumulated output with contribution factors 122

Figures

2.1 Suggested triggers for threat modeling [From SafeCode [16]]. 7

3.1 The overview of method framework. From Johannesson & Perjons [[18], p. 77]. 10
3.2 The identified root causes for the problem. The circles represents main categories 13

4.1 A sequence diagram showing the calculation of the accumulated contribution index and potential trigger 30
4.2 A sequence diagram showing the calculation of PR contribution index. The dotted line for 4.1 and 4.2 indicates how step 4 is done with input from the developer 31
4.3 A sequence diagram illustrating the GitHub workflow 33
4.4 An illustration of the steps of the GitHub action 33
4.5 Screenshot from GitHub of the form containing the questions that was used during the case study 36
4.6 The number of answers for each questions plotted against days into the case study 37
4.7 The contribution indexes for each question and the accumulated contribution index plotted against days into the case study, without contribution factors. 38
4.8 The contribution indexes for each question and the accumulated contribution index plotted against days into the case study, with contribution factors. 39

5.1 How the participants would rate this artifact on a scale from 1 to 5. The data is from both the case study and the normalized data from the two paper prototypes, for both the theoretically described artifact and the implemented one used in the case study. 45
5.2 How useful the participants think the artifact was. The data is from both the case study and the normalized data from the two paper prototypes. The data is from both the case study and the normalized data from the two paper prototypes, for both the theoretically described artifact and the implemented one used in the case study. 46
5.3 How the participants would rate this artifact. The data is from both the case study and the normalized data from the two paper prototypes. The data is from both the case study and the normalized data from the two paper prototypes, for both the theoretically described artifact and the implemented one used in the case study. 46
5.4 The participants of the second paper prototype user test and the case study were asked if they would you use such the theoretically described artifact if it was implemented? 49

5.5 The participants of the second paper prototype user test and the case study were asked if they would be willing to test the theoretically described artifact if it was implemented to see if it is worth it? 49
5.6 How much the participants think this artifact impacts their workflow, on a scale from 1 to 5. The data is from both the case study and the normalized data from the two paper prototypes. The data is about the theoretically described artifact and the implemented one used in the case study. 51
5.7 How much the participants think this artifact slows down their workflow, on a scale from 1 to 5. The data is from both the case study and the normalized data from the two paper prototypes. The data is about the theoretically described artifact and the implemented one used in the case study. 51
5.8 How much the extra hassle or motivation does the participants think this artifact requires, on a scale from 1 to 5. The data is from both the case study and the normalized data from the two paper prototypes. The data is about the theoretically described artifact and the implemented one used in the case study. 52
5.9 How long time the participants reported that it took to fill out the questions during pull requests, from the case study. 53
5.10 How relevant the participants found the questions from the artifact in relation to the code base they were working on, on a scale from 1 to 5. The data is from both the case study and the normalized data from the two paper prototypes. 53
5.11 How many participants understood all the questions from the pull request forms. The data is from both the case study and the normalized data from the two paper prototypes. 54

A.1 The demonstration 10 categories, with randomly generated numbers. Category 2 and 8 have a number between 5 and 10, while the rest has between 1 and 5. Then numbers were generated with the "RANDBETWEEN" function in excel. 70
A.2 The demonstration random values for 30 PRs. The contribution index for each category is calculated by using the "RAND", "RANDBETWEEN", and ROUND functions. While the Date is generated by using "RANDBETWEEN". The last column is there to demonstrate the sum of changes for each pr is 20 or lower. 70
A.3 The demonstrated calculation for the contribution factors of each category for each category, and the calculated PR contribution index value (yellow) and accumulated contribution index (red). 71
A.4 The demonstration accumulated contribution index plotted against time, with a threshold of 1000. 72

C.1 Screenshot of the paper prototype used in the user tests 88
C.2 SUS form used for the evaluation [From John Brooke [28]] 111

D.1 Screenshot from GitHub of the form containing the questions that was used during the case study 113

Tables

3.1 Functional requirements for the artifact 15
3.2 Non-functional requirements for the artifact 15
3.3 The requirements for the method artifact linked to their inspiration 16
3.4 The meta-data interview questions asked before the user tests 18
3.5 The interview questions from the user tests, asked after then user completed the tasks . . . 19
3.6 The interview questions from the user tests, about the theoretical artifact 20
3.7 Linking between interview questions and what they evaluate 21

4.1 Code changes that could be used for triggering threat modeling 27
4.2 Suggested questions that developers can answer if the artifact is not automated. Note that this list is not exhaustive and a suggestion. 29
4.3 An illustration of the structure of the database schema used in the case study 34
4.4 The total number of changes for each questions gathered during the case study 39

5.1 The evaluation goals linked to data generation 42
5.2 Non-functional requirements for the artifact linked to their evaluation 43
5.3 The SUS scores from the two user tests, combined sus score for both user tests, the case study, and the combined SUS score. It is indicated with average, median, and mode, rounded to nearest whole number. 48
5.4 The results when the eight participants were asked how long time they on average spent answering the questions inside GitHub. 49

D.1 The contribution factors used to calculate the accumulated contribution index. 122

Chapter 1

Introduction

This thesis presents and evaluates a new method-artifact that aims to improve software security by indicating when it is time for a new threat modeling session in agile development.

The cost of cybercrime and the spending on cybersecurity is estimated to have surpassed 1 trillion USD in 2020, which was more than 1 % of the global GDP, and almost double that in 2018 [1].
In addition, the Cybersecurity Ventures[1] states that: "*Cybercrime is the greatest threat to every company in the world, and one of the biggest problems with mankind*" [2]. The Norwegian National Security Authority[2] reported in their 2021 annual risk report that the amount of registered severe security incidents has tripled between 2019 and 2020. [3]. In addition, they indicate that the risk has increased with the increased complexity of the software systems and the longer value chains [3].

Software systems have vulnerabilities as a result of flaws in the software architecture [4]. The cost of fixing a flaw or bug increases if it is handled later in the development process [5]. Fixing a flaw during implementation is estimated to be 6.5 times as expensive as during design [5].
An attacker can exploit vulnerabilities in a system to cause damage and harm, but if these vulnerabilities are found and handled, the system can resist threats against these vulnerabilities [6, 7].

Threat modeling is a software security activity that can detect these flaws, and it can be performed early in the software development life cycle [7]. It is therefore important to perform threat modeling. But, security activities such as threat modeling has proven difficult in agile development [6–10]. The current body of research has not addressed the challenges of integrating threat modeling into agile development [6, 10, 11].

Last semester, V. Ahmadi and I performed a pre-project [11] as an extension of a systematic literature review performed by K. Tuma in 2018 [12], where we analyzed 22 threat modeling techniques [11]. One of the main results was a large gap between academia and the industry, especially when it came to the practical details of the threat modeling techniques [11]. Only three of the 22 publications about threat modeling techniques included any details about when and how to perform the threat modeling and how to incorporate the threat modeling with any software development life cycle [11].

[1]https://cybersecurityventures.com/
[2]https://nsm.no/

There are multiple different challenges when integrating threat modeling in agile development, and this is discussed further in chapter 2. One of the problems is determining *when* the threat modeling should be performed in agile development. The current academic body of literature is not clear on this and mainly recommends to shift-left and mitigating the flaws before they are implemented [6, 13]. There exists other recommendations as well: perform threat modeling whenever a new feature is released [14], for every user story [6], periodically [15], when the architecture changes [6], or based on certain specified events [16]. Most of these are not based on the actual code changes, or they are nearly impossible to do in an agile setting, or they are unstructured. This is further discussed in chapter 2, but it is clear there is a great need for more research.

To make matters worse, there is a massive demand for security experts, and it is estimated that there were 3.5 million unfilled positions for security experts worldwide at the end of 2021 [17]. Security experts are the people in the software development companies with the most security knowledge, and they are necessary to support the developers and architects [7]. Among their many skills, they are experts at threat modeling, and they can brainstorm different potential threats, and attacks [7].

Therefore, it is clear that there is a gap in the current research when it comes to integrating threat modeling into agile development. This thesis has the goal of contributing to filling this research gap by creating an artifact that can assist agile development companies with determining *when* the threat modeling should take place based on the actual code changes.
This project follows the design science method framework proposed by Johannesson and Perjon [18], and the result is, therefore, an artifact and knowledge about it. The artifact has been evaluated with two user tests at the IT company of a Norwegian bank *SpareBank 1 Utvikling*, and with a case study at the software development company *Visma*.

More specifically, this thesis tries to answer the following research questions:

1. RQ1: How can threat modeling be triggered based on code changes?
2. RQ2: What is the effect of the suggested approach from RQ1?
    a. To what extent is the artifact effective at solving the problem?
    b. What was the usability of the artifact?
    c. How well did it fulfill its requirements?
    d. How is this artifact compared to other similar artifacts?
    e. What side effects does the artifact have?

The rest of this thesis is structured as follows: a method section that introduces the research method and how step one and two of the design science research was performed, one section about the design and implementation of the artifact, followed by a chapter with the results of the evaluation of the artifact, and lastly, a discussion and conclusion.

Chapter 2

Background

This chapter summarizes related theories and works that are important for the motivation and research for this thesis.

2.1 Threat Modeling

Threat modeling can be defined in multiple ways. Izar and Coles define it as a process where a system is analyzed for weaknesses introduced from less-desirable design choices [6]. The goal is to identify and handle these weaknesses as early as possible before they are implemented as part of the system [6]. Adam Shostack gives another definition of threat modeling as creating a model of a system and figuring out what could go wrong by performing an analysis process [7].
Threat modeling is considered one of the most essential activities to develop secure software systems [4, 7]. If weaknesses in the system are left unhanded, they can be exploited by attackers to cause damage and harm to the system [6, 7].

There exist multiple different threat modeling techniques that have been proposed over the years. These techniques have different focus areas, input types, and other attributes [6, 7, 11, 12]. Some of the most popular and most known threat modeling techniques include: STRIDE, TRIKE, LINDUNN, and SPARTA [6].

2.2 Pre-study on Threat modeling

Last semester, I performed an extension of a systematic literature review by K. Tuma [11, 12], together with a fellow master's student in computer science, V. Ahmadi. In the original SLR, Tuma analyzed and categorized 26 different threat modeling techniques published before 2018 [12]. The SLR was extended by increasing the data range up to 1. Jan 2021, and add a focus on agile development, which led to an additional 22 techniques [11].

The main output of the extension was a more up-to-date catalog of "all" the threat modeling techniques published up to 1. Jan 2021, but we also had some other interesting findings [11].

We noticed that only three of the new 22 threat modeling techniques had suggestions on how to integrate the techniques into a software development life cycle [11]. In addition, several publications did not include any form of practical description of how the technique should be performed [11]. Only 14% of the techniques included any details on how to update the models and analysis if threat modeling is performed iteratively, like in agile development with changing requirements and architecture [11]. In other words, most of the research was very theoretical and did not study or mitigate any of the practical challenges related to threat modeling but was more focused on other goals [11]. Another result was there there were poor connection between the threat modeling and the source code, both for models and the results, and this was proposed as a future work [11].

2.3 Threat modeling in agile development

Threat modeling and other security activities are challenging in agile development [6–10]. Threat modeling is still immature within the agile context, and this is an area that is currently under research. However, we are still missing a deeper understanding of both the challenges and their solutions [6, 10, 19]. One of the challenges is the difference in velocity between the traditional threat modeling in waterfall and the rapid, agile development and how threat modeling can be performed without slowing down everything else [6]. In addition, how can the threat models be kept up to date as the system rapidly evolves [6].
Adam Shostack reminds us that threat modeling is not an inherently heavy and waterfall process, but many of the proposed processes are [7]. In addition, threat modeling might take time, but so will fixing security problems later in the development process [7].

A systematic literature review by Oueslati et al. [20] identified challenges of developing secure software with the agile development approach, the ones of relevance to this thesis are summarized below. The SDLC does not integrate security activities such as risk assessment, and some of the security activities must be repeated at each iteration [20]. In addition, the iterations are short, and it is challenging to find available time for security activities, as these activities are often time-consuming [20]. The incremental nature of agile makes it challenging to complete security assurance activities and to track security objectives, and requirements [20]. Finally, making software secure increases the required development effort, leading to increase costs, and this makes companies compromise security over shortened release time [20].

This year, Bernsmed et al. [10] published a study that combined the results of four previous studies on how threat modeling was performed and adapted in agile development. They found several challenges and findings for threat modeling within the agile contexts, such as: developers are performing threat modeling, threat modeling is not fully integrated as part of the daily activities, the developers lack motivation for performing threat modeling, and it is time-consuming [10]. They report that these findings are similar and confirm many of the other challenges found in the literature [10]. In addition, they discuss that one of the most difficult challenges is to make threat modeling an integrated part of the SDLC [10]. Their findings are similar to one of the main findings from the SLR performed by Oueslati et al. [20]. They also found that threat modeling is a compliance activity for most of the development teams and that the teams don't see that it improves the security, which leads to worse motivation [10].

Multiple solutions are proposed to mitigate these challenges, such as: creating specific SDLCs with security in mind, adapting existing threat modeling approaches to agile, creating new custom threat modeling approaches for agile, and also creating automated tools [6, 7, 12]. Another suggestion is to create a continuous threat modeling methodology (CTM), as proposed by Tarandach and Coles [6]. The CTM is a theoretical threat modeling approach and is based on a set of guideline principles [6]:

- A product team knows their own system better than security experts external to the team
- One cannot expect a team to halt their development to perform threat modeling
- The quality of the analysis will improve as the individuals gain experience
- The threat model must reflect the current state of the system
- Each iteration of threat modeling should improve the threat model
- The findings must be useful and match the current system

Based on these guidelines from Tarandach and Coles [6], one could instantiate a continuous threat modeling approach [6]. One such continuous threat modeling approach is the Autodesk continuous threat modeling approach [6]. This will be described in the next subsection.

Let us assume that all the mentioned challenges above were fixed; there would still be the problem of *when* to perform the refactoring or updating of the threat models in agile development. The next subsection will take a deeper look at this challenge.

2.4 Triggering threat modeling

A fundamental question is, "when should threat modeling be performed in agile development?". In general, the software security literature recommends shifting-left and performing threat modeling as early as possible to mitigate flaws before they are implemented [6, 13], which means that it can be performed as soon as the architecture is ready. The threat model should be updated if necessary later in the development [21]. It is, therefore, possible to perform threat modeling once the initial architecture is constructed, but it will need to be updated as the implementation and architecture changes [6, 14].

In agile development, an initial architecture is proposed by the architects, and the developers use that architecture until it is refactored when the technical debt becomes too great [22]. The code can drift away from the architecture in multiple different ways during development [22]. Two common reasons are due to implementation challenges, and updates to the code [22]. The developers could encounter problems that are challenging to fix, and their solutions cause the implementation to drift away from the architecture [22]. In addition, the architecture could be followed at the start of a project, but as new features and updates are added to the systems, the architecture is not updated to help guide the new implementation or the keep the architecture up to date [22].

How do agile development teams know when the threat modeling needs to be updated? Is it enough to update the threat model whenever the architecture is updated, and what if it never gets updated? Few recommendations and suggestions have been proposed and published, and this subsection briefly summarizes the sources found during the work on this thesis. Many of the publications and recommendations are not academic but instead from grey literature such as blog posts and web pages from security and engineering organizations, because the academic research is scarce in this area.

The **Open Web Application Security Project (OWASP)** writes that "*Ideally, a high-level threat model should be defined early on in the concept or planning phase, and then refined throughout the lifecycle.*" [14]. However, they do not write when this refinement should take place, other than that it should be done after certain events, such as: the release of a new feature, when a security incident occurs, or when an architectural or infrastructural change happens [14]. However, in agile development, new features might be released each day, at least at the end of each sprint [9]. If we follow the definition from OWASP [14], threat modeling will need to take place as often as each day. In addition, it is not always clear when the architecture changes, as the developers make architectural changes when they develop [22].

The **website DevOps.com**[1] recommends that threat modeling is done in the early stages of the software development lifecycle [23]. In addition, they recommend doing it every time there is a change to the systems architecture, after a security incident or a new vulnerability is introduced, and as soon as the architecture is ready [23]. These recommendations are similar to the ones presented by OWASP, and the problem of when the architecture is changed is still present. In addition, when do we know when a new vulnerability is introduced without doing a threat modeling? It is clear that they also see these challenges, as they state that a threat model created today might not be efficient tomorrow [23].

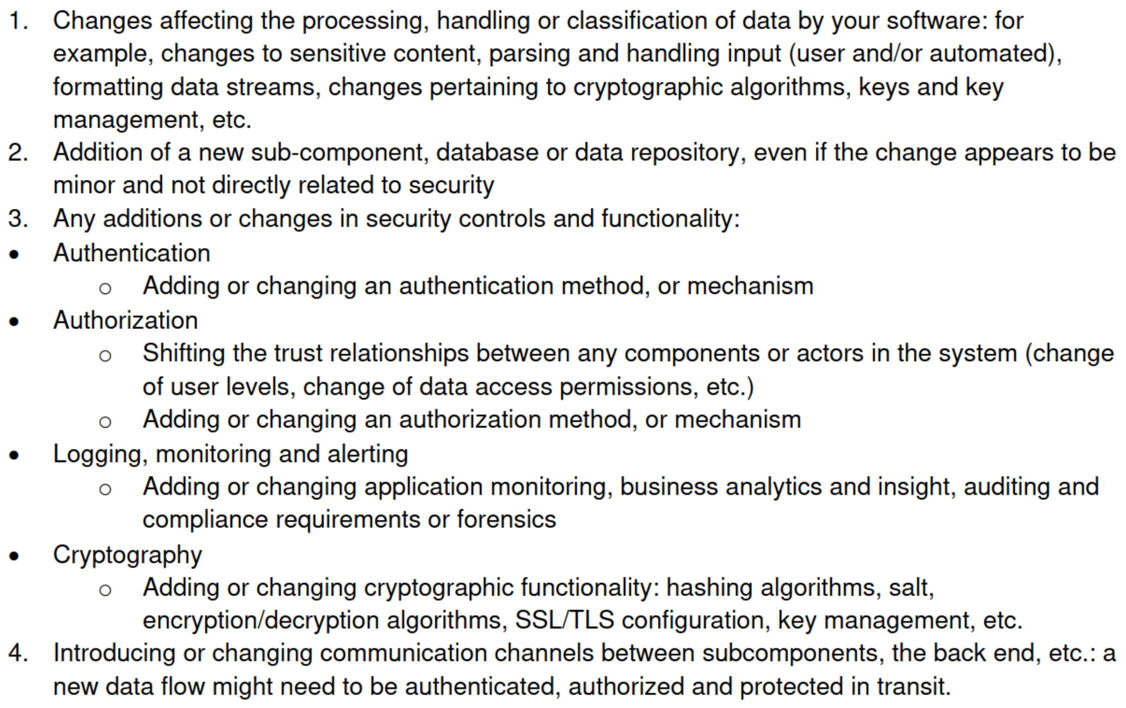
Another recommendation is the **Autodesk continuous threat modeling methodology** [6], as mentioned in the previous subsection. It begins with the team creating a baseline and then performing baseline analysis[6], which is similar to the recommendations presented above. Then, they "threat model every story" to prevent the system from progressing and the threat model from lagging behind [6]. They argue that this challenge needs to be solved at the developer level, as they are the ones making the decisions when implementing, and these decisions impact the architecture and the security of the system [6]. The threat model for every story determines if a story has security value by checking it against the "secure developer checklist" [6]. If the story has security value, it will be marked with a label, and the developer can either mitigate it immediately or mark it as a candidate finding, and then a curator can use this to update the threat model [6]. The developer will follow the checklist to implement mitigations to the threat, which will be documented in the threat model, or mark it as a potential finding [6].

As mentioned in the previous subsection, a study on four different organizations in Norway and how they perform threat modeling was performed by Bernsmed et al. [15]. Among their many findings, they found that all four organizations worked within agile development, and all performed threat modeling at regular time intervals [15]. In addition, they found that threat modeling is time-consuming for the developers; in addition, the developers lacked motivation for performing threat modeling as they felt that it took time away from their "real jobs" [15]. They also found that checklists and clearly defined processes and routines improved the threat modeling [15]. Finally, all four organizations worked within agile development, and all performed threat modeling at regular time intervals.

SAFECode[2] recommends performing threat modeling as soon as the architecture has been established, and can therefore be part of requirements engineering [16]. They state that it is not always clear if the threat model needs to be updated at a given release, but as a rule of thumb: "*changing or adding data that is exter-*

[1]https://devops.com/
[2]https://safecode.org/

nally produced or internally consumed – for example, data stores, log files, webpage contents, descriptive error messages, temp files, etc. – should be considered likely triggers for reconsideration of the threat model." [16]. They have created a list of suggested triggers for threat modeling sessions, shown in Figure 2.1. This list is very interesting as it is one of the first lists of triggers for a threat modeling session based on code changes.



1. Changes affecting the processing, handling or classification of data by your software: for example, changes to sensitive content, parsing and handling input (user and/or automated), formatting data streams, changes pertaining to cryptographic algorithms, keys and key management, etc.
2. Addition of a new sub-component, database or data repository, even if the change appears to be minor and not directly related to security
3. Any additions or changes in security controls and functionality:

- Authentication
    - Adding or changing an authentication method, or mechanism
- Authorization
    - Shifting the trust relationships between any components or actors in the system (change of user levels, change of data access permissions, etc.)
    - Adding or changing an authorization method, or mechanism
- Logging, monitoring and alerting
    - Adding or changing application monitoring, business analytics and insight, auditing and compliance requirements or forensics
- Cryptography
    - Adding or changing cryptographic functionality: hashing algorithms, salt, encryption/decryption algorithms, SSL/TLS configuration, key management, etc.

4. Introducing or changing communication channels between subcomponents, the back end, etc.: a new data flow might need to be authenticated, authorized and protected in transit.

Figure 2.1: Suggested triggers for threat modeling [From SafeCode [16]].

To quickly summarize, there are multiple recommendations and suggestions for when to trigger an update or refactoring of the threat model. However, most are challenging or nearly impossible in agile development and often lack connection to the code or are too un-systematic to ensure security.

Chapter 3

Method

This chapter describes the various methods used in this thesis. Design Science is used as a method framework to guide the work and assure scientific vigor, and the first two steps in the framework are described in detail in this chapter. Interviews, observations, and document analysis are used to generate quantitative and qualitative data.

3.1 Research Methodology

3.1.1 Design Science

Design Science is the scientific study and creation of artifacts both during development and when people use them with the goal of solving a practical problem [18, 24]. People within a practice can utilize an artifact to help mitigate a practical problem that they experience [18, 24]. There are four types of artifacts: *constructs*, *models*, *methods*, and *instantiations* [18]. *Constructs* are concepts, notations, and definitions needed to communicate problems and solutions, while *models* are representations of solutions to problems [18]. *Methods* defines new processes and guidelines for how to solve problems and achieve goals, while *instantiations* are working systems that can be used within practices [18]. This thesis uses design science as a method framework to structure and guide the research. The main contribution is, therefore, an artifact, or more specifically, a new method artifact.

3.1.2 A method framework for Design Science Research

There exist multiple different method frameworks for design science research; this thesis follows the method framework presented by Johannesson and Perjons [18].The framework guide and structure the work, and contributes to the scientific rigor that is required. The framework consists of five main activities: explicate problem, define requirements, design and develop artifact, demonstrate artifact, and evaluate artifact [18]. Design science is always performed iteratively, where the researcher goes from one activity to another in both directions [18]. Each step will be explained in the subsequent sections, and the method is demonstrated in Figure 3.1. It is important to note that the arrows between the steps are not indicating sequence, but inputs and outputs to the steps [18].

Completing all five steps in depth could be a substantial task, and it is therefore common to focus on one or two of the steps while the rests are performed less extensive [18]. This research focuses on the steps: Design & Develop artifact and Evaluate artifact, steps 3 and 5. The main reason is time constraints, as a thorough completion of all steps will take too much time for a single researcher. In addition, I performed a systematic literature review (SLR) last semester, as described in section 2.2, which significantly contributed to explicating the problem and defining requirements. Therefore, it is natural to focus the attention on the design development and evaluation of the artifact.

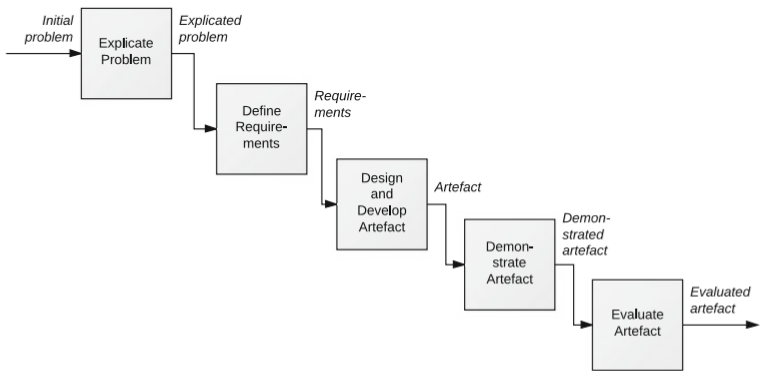


Figure 3.1: The overview of method framework. From Johannesson & Perjons [[18], p. 77].

Explicate Problem
The first step in the design science method framework is to explicate the problem, which is experienced within a practice by a stakeholder [18]. The input to this step is a vague problem. The goal of this step is to define the problem precisely, position it within a context, and ensure that it is of common interest [18], by performing the three sub-activities: *define precisely*, *position and justify*, and *find root causes*. The initial problem needs to be *defined precisely* so that a solution can be found. A more precise definition is generally preferred, but the definition must not be so precise that the solution space becomes too small [18].
Then, the problem needs to be *positioned within a context* so that the practice in which it occurs becomes clear. The problem should also be well *justified*, and it should be clear that it is of general interest [18].
Finally, the *root causes* should be investigated to ensure that the problem is understood correctly [18]. If the root causes are understood, it is easier to create a good solution.
The problem explication is performed at section 3.2.

Requirement Elicitation
The second step of the framework is to draft an artifact that could be used as a solution to the problem and then elicit requirements [18]. The first part is to select what type of artifact should be created: a construct, a model, a method, or an instantiation, as described in subsection 3.1.1. Then, sketch out possible artifacts that could be used as solutions to the problem from step 1. Finally, requirements should be elicited and justified [18]. This is done in subsection 3.2.1.

Design and Develop Artifact
The third step in the method framework is to design and develop the artifact. The goal is to create an artifact that mitigates the problem and fulfills the requirements from step two [18]. The input to this step is the requirements and outline from the requirement elicitation. This step contains four sub-activities: Imagine and Brainstorm, Assess and Select, Sketch and Build, and Justify and Reflect [18]. The researcher needs to generate new ideas or enhance existing ones by: brainstorming, workshops, or interviews. The ideas are assessed against the requirements before one is selected. This artifact is then sketched and built, and the design decisions are justified and reflected upon [18]. This step generates prescriptive knowledge, which can be embedded in the artifacts, and also descriptive knowledge about the design and its rationale [18]. This is done in section 4.1.

Demonstrate Artifact
The fourth step is to prove the feasibility of the artifact by demonstrating its use in one case [18]. The input is the artifact designed and developed in the previous step, and the output is descriptive knowledge about how the artifact works in the test case, but also explanatory knowledge about why it works [18]. There are two sub-activities. The first is to choose or design a case in which the artifact can be demonstrated, and the second is to apply the artifact to this case. [18]. The demonstration can also be seen as a weak form of evaluation [18]. This is shown in section 4.2.

Evaluate Artifact
The fifth and final step is to evaluate how well the artifact solves the problem and how well it fulfills the requirements [18]. The input is the artifact, and the output is descriptive knowledge about the artifact, but also explanatory knowledge [18]. The evaluation has six goals: evaluate how effective the artifact is at solving the problem, evaluate to what extent the artifact fulfills the requirements, investigate the artifact's utility against formalized knowledge, compare the artifact against similar artifacts, investigate its side effects, and finally, identify potential improvements to the artifact [18]. The artifact will be evaluated against five of the six goals, it will not investigate the artifact's utility against formalized knowledge. This is because there are little formalized knowledge about this sort of artifact, and this type of evaluation would not bring much value to this artifact. Instead, the artifacts usability is measured, as poor usability could impact the developers agile workflow more than good usability. The evaluation is done in section 5.2. The six evaluation goals are presented in the list below:

1. Evaluate to what extent the artifact is effective at solving the problem
2. Investigate the usability of the artifact
3. Evaluate the artifact against its requirements
4. Compare the artifact to other similar artifacts
5. Investigate side effects of the artifact
6. Perform formative evaluation to improve the artifact

3.2 Explicate Problem

The problem of *when* to perform threat modeling in agile development is explicated in the chapter 2. However, some of the important points are summarized here, together with the work process for problem explication. The problem explication began last semester as part of the pre-project[11] for this thesis. During this project, it became clear that there was a lack of integration between threat modeling and agile development, and many of the techniques did not take the practical aspects into consideration [11], as discussed in the chapter 2. The main problem is that the threat modeling techniques are not well adapted to the agile methodologies [7, 11], and that the practical aspects of the threat modeling techniques are often ignored in the publications [11]. There are many research gaps that need to be filled, but this thesis focuses on the problem of *when* to update or refactor threat modeling in agile development. The reason is that it seems like the current state-of-the-art literature, presented in chapter 2, has skipped this problem, and it could be an important part of improving threat modeling in agile development. As discussed in the chapter 2, the current literature mainly recommends to shift-left [6, 13], but this is challenging in agile development, as one wishes to reduce the development time and cost [20]. Several other recommendations and suggestions are presented in the chapter 2, but they are often not based on code changes and instead rely on other principles. The problem was further explicated with discussions with my supervisor and security experts from Visma and SpareBank 1 Utvikling.

The problem can be communicated as the gap between the current state and an ideal state. **The current state:** in agile teams, threat modeling is updated whenever development teams feel the need to do it or with timed intervals, such as once a year. The result is a focus on compliance activities, such as threat modeling the entire codebase of a company once a year, but threats could stay in the code without being found for more extended periods of time. **The ideal state:** threat modeling in agile teams is performed when the code has changed to such an extent that the old threat modeling is creating unmanaged risks to the system.

As described in the chapter 2, this problem is rooted within agile development and will therefore be present for most agile development teams. If threat modeling is performed too often, then it could annoy the developers as they will need to spend more time on other activities instead of development, as Bernsmed et al. [10] found that developers feel that it is not their "real" job. But the bonus, of doing it often, is that each threat modeling would be smaller, as there are less changes since last time. But if it is performed too rarely, the threat modeling could take longer. In addition, it could lead to situations where flaws and threats exist in the code for longer periods of time before discovered, if discovered at all, leading to higher risks for the companies [7]. At the same time, there is a massive demand for software security experts [17], as presented in chapter 2. Bernsmed et al. [10] also found that the involvement of security experts had a positive effect on parts of the threat modeling. Therefore, it is important to optimize the use of the security experts and ensure that the threat modeling is performed when it is needed, to not waste their time.
If this problem was solved, the software could be more secure, and we could reduce the workload of developer's and security experts, thus reducing costs. In addition, the developers workplace satisfaction could be increased if we were able to reduce the massive compliance activities.

The root causes for the general problem of threat modeling in agile were identified through the pre-project[11], discussions with my supervisor and security experts at both Visma and SpareBank 1 Utvikling, and through the current body of literature, as discussed in chapter 2. The root causes are illustrated in Figure 3.2, and is explained below.

Developers is a main category and represents the developers that create the software. Today, there is a high demand for developers [25], leading to higher workloads. The developers lack motivation for threat mod-

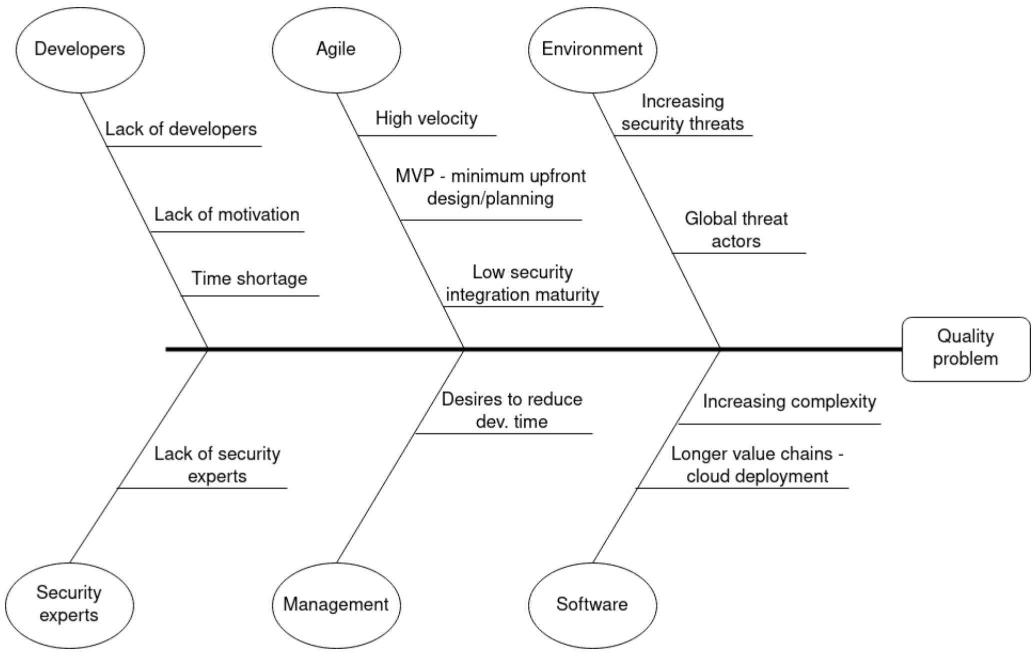


Figure 3.2: The identified root causes for the problem. The circles represents main categories

eling, as they feel that it is not their real jobs and that they don't see its value [10].
Security experts is another main category and represents the people at the companies with the highest knowledge in software security [7]. Today, there is a great lack of security experts, as there is estimated to be 3.5 million unfilled position in 2021 [17]. This means that there is not enough security experts, which is a problem as the literature indicates that threat modeling is better with the involvement of security experts [10].

Agile is a development methodology [9]. It has attributes such as high velocity, minimum viable products, minimum documentation, and iterative workflow [9]. Integrating threat modeling into agile development is challenging [6–10].

Management are the ones who manage the development teams. They are more leaders than developers, and they push the development teams to become more economic, thus reducing the time spent on development activities, in addition they push for shorter development times, which results in a trade off between security and development time [20].

Environment is the environment in which the software exists. More and more software is connected to the internet. In 2021, the Norwegian National Security Authority reported that the number of registered severe security incidents had tripled between 2019 and 2020 [3]. They also reported that the risks are complex and where the threat actors are both governments and criminals [3].

Finally, the *software* is becoming complex with multiple dependencies and the value chains are becoming longer due to more cloud deployment [3], this makes software security more challenging.

The specific root causes for the problem of *when* to perform threat modeling is challenging to determine, as very few academic papers focus on this specific challenge and most literature are grey literature, as discussed in chapter 2. It is therefore assumed that the root causes for the more general problem of threat modeling in agile will have at least some overlap with the specific problem of *when*.

3.2.1 Define Requirements

This thesis presents a new method-artifact to address the problem explicated in section 3.2. The artifact type *method* define processes and guidelines for how to achieve goals and solve problems, they express prescriptive knowledge [18]. The choice fell on a method artifact, as the theory is not mature enough to create an instantiation, which is a working system that can be used within the practise [18]. One could actually view this method artifact as a proof of concept for the future creation of an instantiation. In addition, the available time for a master thesis is not large enough to create an instantiation.

The goal of the new method is to indicate when it is time to perform a new threat modeling session, based on the changes in the codebase since the last time it was performed. This can be achieved in multiple different ways. In the chapter 2 the current body of recommendations and suggestions for when to update or refactor threat modeling is presented. As discussed earlier, many of these are not based on the code changes, but this is one of the goals of this artifact. The recommendations from SAFECode[16] is interesting as they are based on the changes in the code, the recommendations are presented in Figure 2.1. Among other triggers, they recommend threat modeling "*if any changes affect the processing, handling or classification of data by your software* "[16].

The new artifact will trigger threat modeling by detecting and logging code changes, that over time could introduce the need for threat modeling. An index can be introduced to quantify the sum of these changes, and threat modeling should be performed when it reaches a threshold. The changes to the codebase should be detected automatically, but some information might need to be answered by the developers, as they have the best understanding of their own code changes, as indicated by Bernsmed et al. [10]
This artifact is similar to the recommendations from SAFECode [16], but instead of using a single instance of the change to trigger a threat modeling session, it accumulates data over longer periods of time. The reasoning for this is that in agile development, the code changes are often smaller and made in minor increments [8, 9]. This means that the list from SAFECode[16] could trigger multiple times, because of smaller, unimportant changes. Or, a discussion and evaluation will need to take place whenever a trigger from the list happens, to ensure that it is not a false-positive. Another improvement is that the developers will not need to conferee a list whenever they make code changes to look for triggers. Instead, the new method will use accumulated changes to trigger threat modeling. The triggers could be as many as one would like, and they could be tuned based on the risks of the different types of code changes.

The requirements for this artifact were defined by following the principles of requirements engineering as described by Ian Sommerville [8], by iterating the steps: requirements elicitation, requirements specifications, and requirements validation. The requirements were generated and validated with my supervisor and security experts from both Visma and SpareBank 1 Utvikling, until we were all satisfied. The functional requirements (FR) are presented in Table 3.1 and the non-functional requirements (NFR) in Table 3.2.

Table 3.1: Functional requirements for the artifact

ID	Description
FR1	Code changes that could create the need for threat modeling when accumulated should be sorted into types of code changes, and assigned an index factor.
FR2	Whenever developers perform code changes, through pull requests, the contribution index of that pull request shall be calculated by multiplying the number of changes within each category with its index factor and then summarizing this for all types of code changes.
FR3	The pr contribution index for each pull request shall be stored together with a reference to that pull request and the date.
FR4	The system shall continuously calculate the accumulated contribution index in the codebase, by summarizing the pr contribution index of all pull requests.
FR5	If the accumulated contribution index from FR4 surpass a threshold, the team should be notified.
FR6	The accumulated contribution index from FR4 should be reset whenever the team has performed a new threat analysis.
FR7	The team shall be able to change the types of code changes, the index factor for each category, and the threshold.

Table 3.2: Non-functional requirements for the artifact

ID	Description
NFR1	The artifact should affect the developer's agile workflow as little as possible.
NFR2	The types of code changes from FR1 should be meaningful and represent potential dangerous or risky code changes.
NFR3	The developers should be able to handle all interactions with the artifact within the existing tools that they use.
NFR4	If the artifact requires input from a developer, it should be requested using language that the developer understands, without software security jargon or lingo.

FR1-6 is the main functionality of the new method, as described and motivated earlier in this section and covers the core idea. The artifact will detect, store, and accumulate the code changes and use them to trigger threat modeling. The different changes have different factors, as some changes are more dangerous than other. A formula can be used to calculate an index which will represent these changes, and if it reaches a threshold, the team will be notified of the trigger. When the treat modeling has been performed, the team shall be able to reset the value. To make the artifact more useful to the The list of types of code changes, the contribution factors, and threshold should be modifiable for the teams that uses this artifact.

This modifiability will make the artifact more more useful as it can be tailored to different teams. FR7 captures this modifiability.
The functional requirements are shown in Table 3.1, and the link between the requirements and their inspiration is shown in Table 3.3.

Table 3.3: The requirements for the method artifact linked to their inspiration

ID	Inspiration
FR1-FR6	SAFECode[16] and my own innovation
FR7	My own innovation
NFR1	The lack of developers [25], the book Software engineering Global Edition [9], the study by Oueslati et al. [20], Sparebank 1 Utvikling, and Visma.
NFR2	SAFECode [16] and my own innovation
NFR3	The study by Bernsmed et al. [10]
NFR4	Security experts at Sparebank 1 Utvikling, the study by Bernsmed et al. [10], the Autodesk continuous threat modeling [6], Beyond Human computer interaction [26], and security experts from Visma.

NFR1 is the most important non-functional requirement, as this new method should not violate the agile principles nor ruin the benefits of agile development. It is my opinion, based on the results from the pre-study [11], that many threat modeling techniques and other security activities have challenges in agile development because they goes on chore with the agile development, and by adopting the techniques the teams sacrifices their agile values and all its benefits. Therefore, the artifact should affect the agile workflow as little as possible. This requirement also captures that any interaction the developers have with the artifact should be as short as possible. There is a lack of developers [25], and an important goal of agile development is to reduce the development time [9], therefore it is important to require as little time as possible. The study by Oueslati et al. [20] also identified that security activities are time consuming, which makes companies compromise security to reduce development time. It is therefore imperative that this requirement is fulfilled. Security experts from both SpareBank 1 Utvikling and Visma also stressed the importance of this requirements.

The main idea behind this artifact, as discussed earlier in this chapter, is to use different types of code changes to trigger threat modeling. These types of code changes must be meaningful and represent potential dangerous and risky code changes to be able to be used as triggers, much like the recommendations from SAFECode [16]. This is captured by NFR2

NFR3 is about the convenience of the artifact seen from the developers. Bernsmed et al. [10] found that the developers was lacking motivation for performing threat modeling, and one can assume that the same can be said about their motivation for an artifact that can trigger treat modeling [10], and it is therefore important that it is not an inconvenience for them.

NFR4 is also partially inspired from the study by Bernsmed et al. [10] as an interaction on the developers terms and notation will make it more convenient and smoother for the developers. The Autodesk Contin-

uous Threat Modeling methodology [6] uses a if-this-then-that checklist that the developers will interact with. The checklist is concise and it is written in the language that developers use [6]. One example of this type of language is "*...added web (or web-like, REST functionality)*" from [6]. Human computer interaction theory stressed the point that the interfaces musts be designed with the users in mind to increase the usability [26]. It is therefore logical to use "developer-oriented language" when the users are developers. The security experts from both SpareBank 1 Utvikling and Visma stressed the importance of this. The non-functional requirements are shown in Table 3.2, and the requirements linked to their inspiration is shown in Table 3.3.
It is possible to create multiple additional requirements, both functional and non-functional, but these are not essential for this method artifact, and will instead be presented as future work in section 7.1, togheter with other future work.

3.3 Data collection methods

In this thesis, data is generated in three different ways: two user tests with SpareBank 1 Utvikling, one case study at Visma, and an ex ante argument. In addition, expert discussions have been performed throughout the research to gain confidence and feedback from the industry.

The first user test was used as a formative evaluation during the first iteration of the design. The second user test and the case study at Visma were done during the second iteration. This means that the first user test is not directly comparable to the second user test and the case study, as they have tested slightly different artifacts. At the same time, the designs were similar, and some comparison is therefor possible.

3.3.1 User testing

User testing is used to generating data for this thesis. The user testing consists of usability testing, as well as structured interview questions to gain more knowledge about the artifact and to evaluate the different goals in step 5 of the method framework, as presented in Figure 3.1.

Usability testing is used to generate information and knowledge about the artifact. ISO 9241-11:1998 defines usability as "*extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*" [27]. It defines effectiveness as "*Accuracy and completeness with which users achieve specified goals*", efficiency as "*resources expended in relation to the accuracy and completeness with which users achieve goals*, and satisfaction as "*Freedom from disfavored, and positive attitudes towards the use of the product*" [27]. ISO 9231-11:1998 also defines context of use as "*users, tasks, equipment (hardware, software and materials), and the physical and social environments in which a product is used*" [27]. It recommends that effectiveness, efficiency, and satisfaction are measured with at least one measure each [27].

The goal of the user testing performed in this thesis is, therefore, to measure the usability and other aspects of the artifact. The questions asked during the user testing can be tied to one of the six goals of the evaluation step as presented in step five in Figure 3.1. At the end of the usability tests, the subjects filled out a system usability scale (SUS) form, which provides a quick and dirty metric for usability [28]. The SUS form is a standardised form for measuring the usability of a product, but it has low quality, and it is important to not put too much emphasis on the score [28].

The user tests were paper-prototype tests, which means that the subjects interacted with a paper version of the system, while the functionality was mimicked by the researcher [26]. This was chosen as it is an inexpensive way of performing tests, and it was a good way to test a method artifact, as it does not require a finished implementation.
The tests followed the principles of user testing [26], and they therefore began by introducing the users to the test setup, explaining the equipment and goals of the test, and that they as persons were not being tested and that they could abort at any time. The full test guid is in Appendix C. Each participant read the consent forms from NSD, and gave their informed consent. The subjects were given a quick introduction to the artifact, its motivation, goals, and how it worked. The subjects were background questions to gather some information which the answers can be seen in the light of, shown in Table 3.4. Then, they were observed and timed while performing a task. After the observation, they were asked questions about the task they performed, these are shown in Table 3.5.

Table 3.4: The meta-data interview questions asked before the user tests

ID	Meta Data Question	Answer type
1	How long have you been working as a developer/tester.	less than 1 year, 1-2 years, 3-5 years, 6-10 years, more than 10 years
2	Rate your own security skills.	scale: 1-5
3	What is your primary work title.	Developer, tester, architect, designer, manager
4	When you develop, are you more focused towards features or security?	scale: 1 - 5 (features)-(security)

The artifact is a method, and the user tested a paper prototype of an instantiating of this method, which means that the results of that test measures the usability and usefulness of that instantiating, instead of investigating the artifact itself. Therefore, the user tests were followed by an interview, where the users got a description of how such a method could be instantiated more ideally than the paper prototype, and then the answered questions about that theoretical instantiating. This makes it possible to gather some data about the method artifact, and not just the paper prototype. It is however important to note that such a theoretical approach will have limited validity, but so will the paper prototyping. The questions about the theoretical approach is shown in Table 3.6. It is important to note that some questions were only asked during one of the user tests, this is the case for questions 17-21. The reason is that the the results from the first user test were used decide between alternative questions and form layouts before user test two and the case study. Question 19-21 were only present on the second user test, because it became obvious that they were lacking after the first user test.
Detailed information about the interview guides, descriptions, NSD forms, questions, tasks, and more is

found in Appendix C.

Table 3.5: The interview questions from the user tests, asked after then user completed the tasks

ID	Question	Answer type
5	Do you have any immediate thoughts?	Free text
6	What worked well?	Free text
7	What worked bad?	Free text
8	How relevant were the questions for the codebase you work with?	scale: 1-5
9	Did you understand all the questions?	Yes, No, Unsure

In the first iteration of user tests, all the questions where the participants were asked to give a rating, they were asked to use a scale from 1 to 10. However, during these tests, it became clear that this scale felt confusing and overwhelming for several participants, as the range was large, and they had problems with determining what to rate. therefore, the second iteration and the case study used a scale from 1 to 5. This creates a problem, as these scales are not directly comparable. To allow for simple comparisons, we convert the scale from 1-10, to a scale from 1-5, using linear transformation. This introduce greater uncertainty to the results, but the advantages of gaining a larger sample makes it worth it. Detailed information about this transformation can be found in Appendix C.

The goal of the tests are to evaluate six different goals, as described in subsection 3.1.2. Each interview question is linked to one or more of these goals, this is shown in Table 3.7. In addition, the time they take to complete the task is measuring the efficiency and NFR3. The open ended questions, 5, and the notes from the observation is not part of this list, but they are formative evaluating the artifact to improve the artifact.

SpareBank 1 Utvikling is the software development company for the SpareBank 1 alliance of banks in Norway. They are part of the fintech sector, and it is therefore possible that security has higher priority than in other sectors. It is important to note that while this is a bank, they are highly agile, and they deploy between 30 and 60 times a day. They follow the principles of Google OKRs and monday commits-Friday wins, and more. This means that any testing at this company is highly relevant for the agile focus of this thesis. This company were chosen because I will begin working in SpareBank 1 Utvikling after the thesis, and it was therefore easier to get in contact with them and recruit developers to participate. Special care have been taken to ensure the correctness and independence of the results, as discussed in section 3.5.
The user tests were tested at two different development teams. The first user tests were tested with a team that primarily worked with full stack development, where both front end and back end are in focus. The second team was a more back end oriented team.

Table 3.6: The interview questions from the user tests, about the theoretical artifact

ID	Question	Answer type
10	Range such an artifact.	scale: 1-5
11	How much would this affect your workflow?	scale: 1-5
12	How much would this slow down your development?	scale: 1-5
13	How much hassle/motivation would this new artifact require of you?	scale: 1-5
14	How usefully would such an artifact be?	scale: 1-5
15	Do you think it would be worth the additional hassle/motivation?	Yes, No, Neither, Unsure
16	Do you think it would change the size of your PRs?	Larger, Smaller, Neither, Unsure
16	Would you use such an artifact if it existed?	Yes, No, Neither, Unsure
17	Did you like that the questions were grouped after new and modified/edited and then the questions themselves are shorter (ALT A), or would you have them all written out fully (ALT B)? **(Only asked on the first user test)**	Alt B, Alt A, Unsure
18	Would you prefer if the questions were inside GitHub or BitBucket and with the type of format and answer type as this paper prototype test, or would you prefer to be sent to an external site with better formatting and "smoother" answering mechanisms? **(Only asked on the first user test)**	Inside GitHub/ BitBucket, external, unsure
19	Would you use such an artifact if it existed? **(Only asked on the second user test)**	Yes, No, Neither, Unsure
20	Would you test such an artifact to determine if it is worth it? **(Only asked on the second user test)**	Yes, No, Neither, Unsure
21	Do you think this would be better or worse than similar security artifacts? **(Only asked on the second user test)**	Better, Worse, Neither, Unsure

3.3.2 Case study

A feature team at Visma tested an mvp implementation of the method-artifact as a GitHub workflow for three weeks to evaluate the artifact in more detail. The implementation is discussed and described in section 4.3, and it closely resembles how a team of developers will experience the new method-artifact, if it is not automated, and can therefore be used to study the usability and usefulness in more detail during the user tests. But, the developers are not experiencing the benefits of this new approach, only the extra work they have to do, which could impact the evaluation. In addition, the developers answers can be used for data analysis, as they generate data whenever they create PRs. This data is similar to the data the system will either generate itself, or the developer will input. This means that the analysis of this data can be used to evaluate the method itself. The participants were asked the same theoretical questions about the theoretical artifact as in user tests, presented in Table 3.6. In addition, the same type of data was gathered about the implemented artifact, by slightly rewriting the theoretical questions. Since these questions are

Table 3.7: Linking between interview questions and what they evaluate

Question ID	Evaluates
6	Formative evaluation.
7	Formative evaluation.
8	FR2.
9	NFR4.
10	To what extent the artifact is effective for solving the problem.
11	NFR1.
12	NFR1.
13	NFR1.
14	To what extent the artifact is effective for solving the problem.
15	To what extent the artifact is effective for solving the problem.
16	Investigate side effects.
17	Form layout (not any of the six evaluation goal).
18	Form layout (not any of the six evaluation goal).
19	Usability.
20	Usability.
21	Compare artifact to similar artifacts.

almost identical, they contribute to the same evaluation goals, and are therefore not presented in identical tables as for the user tests.
Detailed information about implementation they used are found in section 4.3, and detailed information about the test-setup can be found in Appendix C.

The case study were overseen by the security team at Visma, and performed by one of the feature teams. It consisted of 9 people: 1 Product Owner, 1 Test Engineer, 1 Infrastructure Engineer, 1 Architect, 5 developers. They are working agile, by following the principles of scrum, with 2 weeks sprints and the the normal ceremonies. This team of Visma was chosen, as the supervisor for this thesis had relations with the team leader.

After the case study were done, an unstructured interview was performed with the head of security development, which is the leader of the security team and one of the senior infrastructure engineers. In this interview they were shown the raw output of the artifact, some simple analysis as well as the accumulated contribution index and its actual output, as described in detail in Appendix C. The goal of this interview was to gain a better understanding of the usefulness of the artifact and how it was perceived by security experts and management.

As a student, I was outside the Visma organization, and had therefore no access rights to their code or other resources, because of security and industry reasons. I was therefore not able to directly interact with the participants during the case study, and I could not observe the artifact during use. Instead I got the raw output of the artifact weekly and when the case study was finished.

3.3.3 Expert discussions

Throughout this thesis I had regular meetings with security experts from multiple different companies. These meetings were used to gain confidence and feedback from the industry, about the the problem of triggering threat modeling and the proposed artifact. Many of the features of the artifact and the recommendations stem from these discussions. But most importantly, they have provided confidence in the proposed artifact. Not all experts are mentioned by name, but their workplaces included: SpareBank 1 Utvikling, Visma, Twilio inc., Jemurai, and DNV.

One of these experts were Matt Konda[1], which foundad Jemurai, a security consultant company. He has done much work in the area of software security and threat modeling. He has worked on multiple different companies, among them, the OWASP Glue Tool project[2]. Another expert is Rohini Sulatycki[3], which is a Security Architect at Twilio inc, and the serves as the current chapter lead for OWASP South Florida.

The security experts from SpareBank 1 Utvikling include: the two security architects Jon Are Rakvåg[4] and Adrian Alexander Eriksen[5].

The security experts from Visma include: Monica Iovan[6] which is the head of security development, and the two senior Infrastructure Engineers Nicoleta Botosan[7] and Romina Druta[8].

3.4 Data Analysis

The goal of data analysis is to investigate the data to determine patterns and draw conclusions [29, 30]. Both qualitative and quantitative data are generated from the data generation. This section describes how was analyzed.

Quantitative Data Analysis

Quantitative data are evidence and data based on numbers [29, 30]. Quantitative data analysis can be performed in multiple different ways, with a wide range of techniques, ranging from simple to advanced [29, 30]. The simplest form of analysis is done with graphs, tables and charts, while more advanced techniques use descriptive statistical methods, and the most advanced use complex statistical techniques. The advantage of complex statistical techniques is that they can analyse the results to determine that patterns are

[1]https://www.linkedin.com/in/mattkonda/
[2]https://owasp.org/www-project-glue-tool/migrated$_{c}ontent$
[3]https://www.linkedin.com/in/rohinisulatycki/
[4]https://www.linkedin.com/in/jonarer/
[5]https://www.linkedin.com/in/adrianeriksen/
[6]https://www.linkedin.com/in/monica-iovan/
[7]https://www.linkedin.com/in/nicoleta-botosan-0aa356116/
[8]https://www.linkedin.com/in/romina201/

simply not a random coincidence [29, 30]. The disadvantage is that the researcher needs to be competent with the techniques, or the analysis could be faulty.

There exists four main types of qualitative data: nominal data, ordinal data, interval data, and ratio data [29, 31]. *Nominal data* describes categories and cannot be used for analysis other than the frequency of each category [29, 31]. One example of nominal data would be if subjects were asked if they prefer cats or dogs.
Ordinal data are numbers placed on a qualitative scale, this allows for simple data analysis, but one cannot know the difference between the ranks [29, 31]. An example of ordinal data is when a subject is asked to rank something on a scale, and each answer represents a number: strongly disagree(1), disagree(2), unsure(3), agree(4), strongly agree(5).
Interval data is similar to ordinal data, but it is measured against a quantitative scale where the difference between each point on the scale is similar, meaning that the entire scale is proportionate [29]. This data can be added and subtracted during analysis, but not multiplied or divided [29, 31]. An example is years, it is the same difference between year 1000 and 1004, as between 2000 and 2004 [29].
Ratio data is similar to interval data, except that it contains a true zero, which is all results are measured against [29, 31]. This means that ratio data can support division, subtraction, multiplication and division, because it has a reference point [29, 31]. One example is peoples age, age can be 0, and all peoples ages are referenced against 0 [29].

Two types of quantitative data are generated in this thesis: from structured interviews with developers and the data generated through the case study tests. The interviews generate both nominal and ordinal data, while the case study generates nominal data. Nominal and ordinal data does not support advanced statistical analysis, and can therefore only be used for simple statistical method to describe the central tendencies: such as mean, median, mode, and frequency [29, 30].

Qualitative Data Analysis

Qualitative data is all data that is not numerical, such as sound, words, images, etc. [29, 30]. Qualitative data analysis involves abstracting information from the data, such as themes and patterns, that has importance for the research [29, 30].

Qualitative data is generated in this thesis through interviews with developers, with questions that are more open-ended than the once's included in the quantitative data analysis. It was also generated through the semi-structured interview with the head of security development and a senior infrastructure engineer at Visma. The method for analysing this data is to identify key themes and create segments, before categorising these segments, then the categories and segments are refined [29, 30]. Finally, patterns and other interesting findings are identified [29, 30].

3.5 Ethics

Researchers should conduct their research and behave in an ethical manner, and everyone involved should be treated fairly and with honesty [29]. This thesis primarily collected data through interviews with participants, and it was therefore important to ensure their privacy and protection. Therefore, all the data was anonymized from the very beginning of the project. In addition, this project follows the ethical guidelines from the Norwegian University of Science and Technology [32], and created a data management plan, notification form, and applied for approval from NSD[9]. The NSD application ensures that many of the ethical considerations from the Personal data act [33] is followed, such informed consent and the collection, storage, and processing of data. The notification form was approved on the 31. Jan 2022, and can be found in Appendix B.

The participants gave their voluntary, specific, informed, and unambiguous consent, in a written, documented form, in line with the personal data Act article 32 [33]. The consent forms informed the participants of what their participation entailed, their rights, why they were asked to participate, and more, as recommended by the NSD [34]. The notification forms can be found in the Appendix B.
The data was securely stored on the NTNU Office 365 Sharepoint, as NTNU has a data processing agreement with Microsoft. To ensure additional security, all data was anonymized continuously. The data will be deleted at the end of the project, and none of the data will be archived.

A fundamental principle of science is that researchers must behave with integrity and record data accurately, fully, and honestly [29]. Part of this is to ensure that one is not affected by influences or wishes to present a certain type of result. My supervisor works at both Visma and at NTNU, and I will begin working at SpareBank 1 Utvikling. In addition, the artifact in this thesis is of my own design. Research is about creating new knowledge [29], and the results presented are of equal importance for improving software security, weather they are positive or negative for the artifact itself. I therefore believe that the results presented in this thesis are not biased by the relations mentioned above.

[9]https://www.nsd.no/

Chapter 4

Triggering threat modeling

This section contains step three, design and develop an artifact, and four, demonstrate artifact, in the design science research methodology framework, as described in subsection 3.1.2. It also contains the implementation details of the instantiation used in the case study, and the raw outputs of the case study.

4.1 Design and Develop Artifact

The third activity in the method framework is to design and develop an artifact that fulfils the requirements defined in step two, as described and outlined in subsection 3.2.1. During design and implementation, the security experts from Visma and SpareBank 1 Utvikling, as well as my supervisor, were conferred to improve the artifact. The design was performed in two iterations, and the artifact was evaluated at the end of both, to improve the artifact before the case study.

The inspiration for this artifact and its requirements are thoroughly discussed in step one and two in the design science method in section 3.2 subsection 3.2.1, and the background section chapter 2. The process for designing the artifact was the iterative process of studying the theory, brainstorming, proposing a solution, and discussing it with the security experts and my supervisor.

4.1.1 The design of the artifact

This section presents different potential designs for the artifact. This artifact is a method-artifact, and its goal is therefore to defines a new process and guideline for how to solve problems and achieve goals [18]. This artifact type is theoretical, and its design does therefore not include implementation details, but instead focuses on the goal themselves. But to be of more use to future researchers and developers, some potential implementation details will be explained and discussed.

This artifact needs a way to detect and store different code changes, and the design process therefore began with a brainstorming and categorization of the different types of data that could be used. The most promising data is the raw code changes from the version control systems and the outputs from static code analysis tools. These data could be used to automate the part of the artifact that detects and categorizes the code changes. Another way is to ask the developers to inform the artifact about the different code changes they have performed.

One such approach could be to present them with a questionnaire before they merge pull requests, and that as raw data for the artifact. This is similar to the Autodesk continuous threat modeling approach [6], but this technique does not gather information through the pull requests, it instead used the questions to make the developers aware of security actions they would need to take [6].
The questions for this artifact could instead be answered during sprint planning, which is more in line with the principles of shift-left [6, 13]. Such an approach would have the advantage of being able to trigger threat modeling before the code is implemented instead of after the implementation has happened, but it also has its downside. It will not be based on the actually code changes, but instead on the planned code changes. It is possible that the implementation takes another direction than the planning, or that the implementation is challenging and that the developer must take decisions that were not planned, and therefore will not be captured by the artifact. One can also imagine a situation were a feature is dropped during implementation or other larger changes are made, as changing requirements is common in agile development [9]. One final downside is that such an approach cannot be automated, and will require additional work from the developers, compared to a method that uses the code changes and the outputs of static tools. This is not ideal, as sprint planning meetings should be as short and concise [35], and it could quickly become a substantial task if the developers were to do this together for all the features they planned to implement in a sprint. Therefore, automatic data gathered from version control software or a manual questionnaire during pull requests seems most ideal.

Some of these code changes create more urgent need for threat modeling than others, such as a change in a data class versus a change in the core authentication functionality. It is therefore a need for some factors to make sure that this difference is present in the triggering functionality. Therefore, each code change should have a contribution factor to represent this difference. This factor will be different for each system, but one can assume that the general trend will be that code with traditional security value will have highest factors.

It is challenging to come up with a list of suggested types of code changes which this artifact could use as input, because the literature is scarce in this area, as discussed in the chapter 2. One list of suggested triggers based on code changes were found during the work on this thesis, it is published by SAFECode [16] and shown in Figure 2.1. Their rule of thumb for when threat modeling should be triggered is "*changing or adding data that is externally produced or internally consumed...*" [16]. For certain parts of the code, it might be necessary to perform threat modeling whenever any changes are introduced, such as the core authentication mechanisms. But at the same time, many code changes are small in agile development [9], and a single code change might not be reason enough to perform threat modeling. It is possible to imagine an agile development team that ends up with new triggers every day, because of small, non-security-related code changes.

The list from SAFECode[16], is therefore a good starting point for potential triggers, but this artifact should also look for code changes that are not dangerous or risky by themselves, but could be if enough changes are introduced. This could be code changes that if accumulated over longer periods of time causes drift in the architecture, which is when the architecture of the systems differs from the planned architecture [22]. These code changes could be sorted into categories. The changes within each category could be counted to calculate the contribution from each category towards the need to perform threat modeling. A formula could add the contributions from each category or type of code change and represent this with an index that represents the need for threat modeling. The different types of code changes could have different factors depending on their contribution to the need for a threat modeling. Another approach is to evaluate how big percentage of code lines are changed within a certain method since last threat modeling.

As mentioned in the chapter 2, multiple different sources recommends to perform threat modeling when the architecture is updated, this artifact could trigger threat modeling based on the actual changes in the architecture of the system, by detecting and logging the code changes.
This artifact can therefore trigger threat modeling based on code changes. By using the factors for the different types of change, some code changes could only need to happen once to cause a trigger, while others needs to be accumulated before they cause a trigger.

Table 4.1: Code changes that could be used for triggering threat modeling

ID	Description	Source
Type 1	Changes that affect the processing, handling, or classification of data by your software	From SAFECode [16] trigger 1.
Type 2	Addition of new database or data repository	Inspired from SAFECode [16] trigger 2.
Type 3	Additions and changes in security controls and functionality: authentication, authorization, logging, monitoring, alerting, cryptography.	From SAFECode [16] trigger 3.
Type 4	Addition or changing communication channels between sub-components, the back end, etc.	From SAFECode [16] trigger 4.
Type 5	New classes / modules	Inspired from software architecture theory [22].
Type 6	Changes in access and operations functionality on data storage	Inspired from software architecture theory [22] and discussions with security experts at Visma and Sparebank 1 Utvikling.
Type 7	New third party dependencies	Discussions with security experts at Sparebank 1 Utvikling and Visma, and the 2021 annual report from the Norwegian National Security Authority [3]
Type 8	New APIs and opened ports	Inspired from the Autodesk secure developer checklist [6] and discussions with security experts at Sparebank 1 Utvikling and Visma

A list of types of code changes is shown in Table 4.1, they are created with the assistance from security experts from Visma and SpareBank 1 Utvikling, the recommendations from SAFECode[16], the Autodesk secure developer checklist [6], and software architecture theory [22]. It is important to note that this list is not extensive, but are suggestions for types of code changes that could be used to create triggers.

Code change type 1-4 in Table 4.1 is almost identical to the ones presented by SAFECode [16]. All four seems logical and as there are no other publications with other recommendations, there are no argument against them.
Type 5 is inspired from the software architecture theory [22], and are among the most common types of changes. It is one of the types of changes that by themselves is not a reason for a trigger, but if many enough of changes are made, then it could be necessary.
Type 6 is also inspired from the software architecture theory [22], but also came to be through discussions with security experts from Visma and SpareBank 1 Utvikling. It attempts to capture changes that modify the access of databases or other potentially dangerous changes, such as modifications, deletion, etc. This is similar to type 1, but type 1 is more focused on the processing of data. It is also similar to type 2, but it is more focused towards new storage units.
Type 7 captures the addition of third party dependencies. Security experts from both Visma and SpareBank 1 Utvikling agreed that new dependencies are risky activities, that should not be done without care. In addition, the Norwegian National Security Authority reported that companies should investigate their own dependencies [3].
Type 8 captures changes to the trust boundaries in the form of exposed APIs and ports. This is based on the Autodesk secure developer checklist [6] and security experts at both SpareBank 1 Utvikling and Visma.

4.1.2 Artifact description

This section contains the concrete description of the method artifact proposed in this thesis. The description is based on the discussion about the design in the previous subsection.

A list of the different types of code changes is created, the ones presented in Table 4.1 can be used. Each type is assigned a contribution factor which is used to indicate the difference in risks from changes within the different types. The artifact is connected to the version control system where the code is stored and edited, so that it can either use the code changes directly, use the output of static analysis tools, or rely on input from the developers without them needing to use other tools. The artifact is activated whenever a developer creates a pull request (PR). The code changes are either automatically detected or inserted into the artifact by a developer through a questionnaire. The number of changes within each type of code change is counted and multiplied with the contribution factor for each type, to calculate the contribution from this PR towards the triggering of a new threat modeling session. This contribution is called the PR contribution index. This PR contribution index is stored in a database with a reference to the pull request and the date, which makes more advanced analysis possible. This is illustrated in Figure 4.2.
When the database is updated, async in relation to the developer and the PR, the artifact accumulates the PR contribution indexes to get the accumulated contribution index. This is compared with a pre-defined threshold, and the development team and the related security experts are notified if the value has exceeded the threshold. This is illustrated in Figure 4.1.

The number of code changes within the different types presented in Table 4.1 should ideally be automat-

Table 4.2: Suggested questions that developers can answer if the artifact is not automated. Note that this list is not exhaustive and a suggestion.

ID	Questions	Related type of code change in Table 4.1
Q1	How many modifications or additions have you made to functionality that receives user input?	Type 1
Q2	How many new ways of receiving user data have you added?	Type 1
Q3	How many modifications or additions have you made to functionality that process data?	Type 1
Q4	How many new data are you storing?	Type 1
Q5	How many new databases have you added?	Type 2
Q6	How many additions or modifications have you made to security mechanics: authentication, authorization, or cryptography?	Type 3
Q7	How many new logs have you added?	Type 3
Q8	How many modifications of additions have you added to monitoring or alerting functionality have you done?	Type 3
Q9	How many new integrations have you created?	Type 4
Q10	How many new classes or modules have you added?	Type 5
Q11	How many modifications or additions have you made to the access or operations functionality of data storage?	Type 6
Q12	How many new dependencies have you added?	Type 7
Q13	How many new APIs or opened ports have you added?	Type 8

ically detected, which would make the proposed artifact automated. Automation of the artifact is outside the scope of this thesis, as the goal is to present a method-artifact, not an instantiation. Each type of change could also be represented by one or more questions, which the developers will need to answer whenever they create a pull requests. Suggested questions are presented in Table 4.2. A team that uses this artifact could refine the questions and add other questions that could help detect dangerous changes. The team can also change the threshold and the contribution factors, based on the team and their systems. The questions presented in Table 4.2 were created with help from the security experts at SpareBank 1 Utvikling and Visma.

4.2 Demonstrate Artifact

This section contains step four of the design science method framework, as described in subsection 3.1.1, by demonstrating the artifact created in section 4.1. To demonstrate the artifact one would need input from a development team over a long period of time, such as 6 month, and then analyze and extract data from this data. It could also be possible to use an open repository to gather pull request and then manually classify the code changes, but this would take too much time, and it is challenging to classify code changes without detailed knowledge of the codebase. Therefore, the demonstration will be done with a fictional

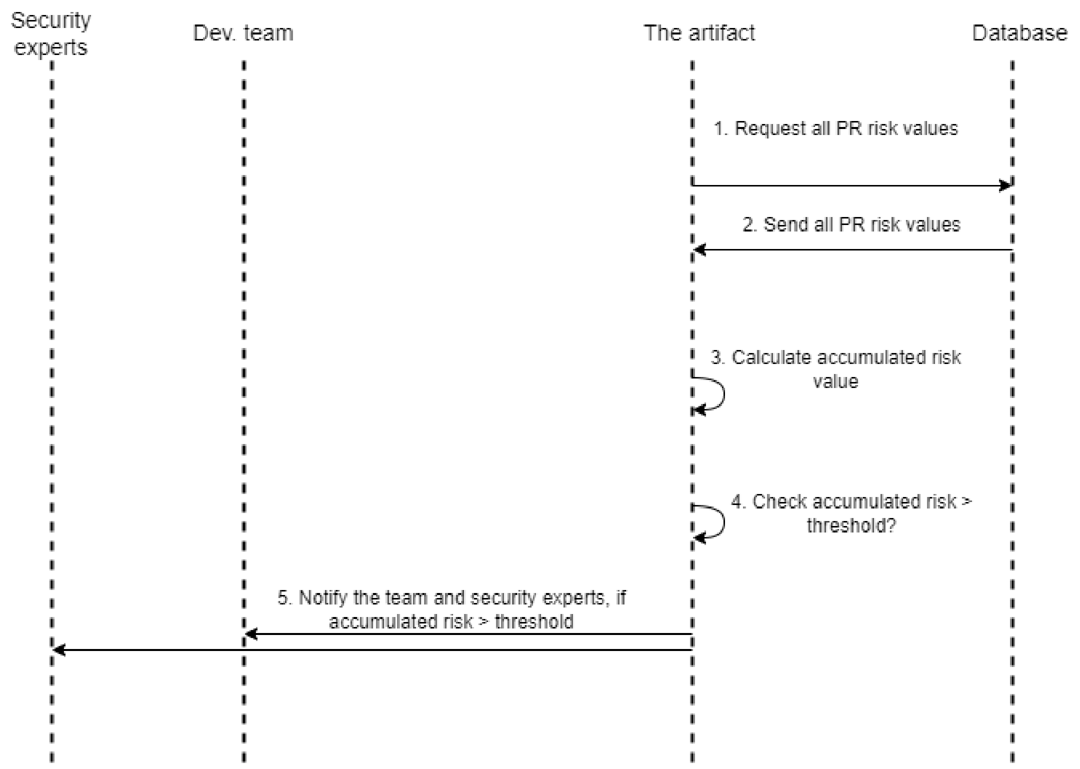

Security experts
Dev. team
The artifact
Database
1. Request all PR risk values
2. Send all PR risk values
3. Calculate accumulated risk value
4. Check accumulated risk > threshold?
5. Notify the team and security experts, if accumulated risk > threshold

Figure 4.1: A sequence diagram showing the calculation of the accumulated contribution index and potential trigger

case. In addition, the artifact have gained confidence through discussion it with multiple security experts, as described in subsection 3.3.3. Finally, the demonstration is also done as part of the case study at Visma.

The proposed artifact is very simple at is core, it accumulates a value until it reaches a threshold. This can be demonstrated, but it provides little information and confidence. At the same time, the artifact is dependant on the types of changes or questions representing the types of changes answered through human interaction. It is challenging to demonstrate this in a fictional case. It is also challenging to demonstrate a method artifact, as it is theoretical.
Therefore, this section will provide a theoretical demonstration of the artifact, while Appendix A contains a demonstration using randomly generated numbers and calculations, for the interested reader.

4.2.1 Theoretical demonstration

We can assume that this artifact has 10 types of code changes. Each type has a contribution factor, based on evaluations of the team using the artifact. The team chooses a threshold. The developers writes code and creates and merges pull requests, and the artifact either categorises the code changes automatically, or the developers answers 10 questions whenever they create a PR. The artifact calculates the contribution index for each PR, and stores it together with a reference to that PR and a date stamp. It continuously calculates

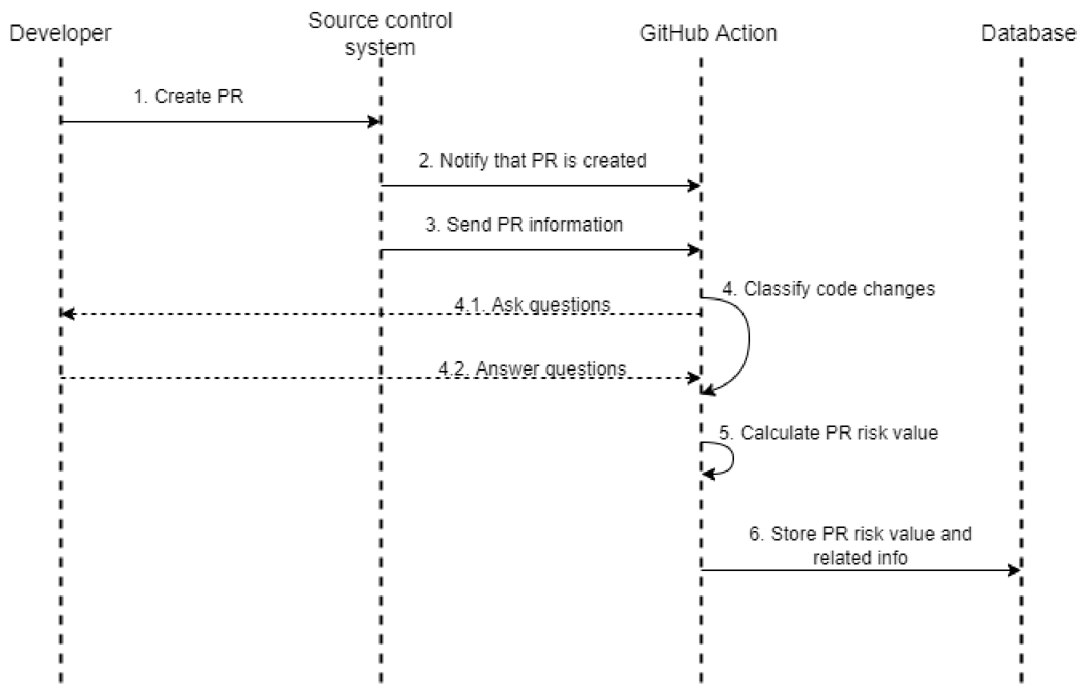


Figure 4.2: A sequence diagram showing the calculation of PR contribution index. The dotted line for 4.1 and 4.2 indicates how step 4 is done with input from the developer

the accumulated contribution index, by summarizing all the PR contribution indexes.
Sooner or later, the accumulated contribution index surpasses the threshold, and the artifact notifies the team and security experts. Together, they decide how to handle the trigger, and determines a course of action. They might perform threat modeling and then reset the accumulate contribution index by emptying the storage. Or they can increase the threshold, if they think it is too low.
This is repeated forever, and the team performs a short, regular meeting, for example once a month, to maintain the questions and factors.

We have now demonstrated the feasibility of the artifact, with a theoretical argument. The theoretical situation above is represented with values in Appendix A, for more information about the different concepts and elements.

4.3 Case study implementation

The artifact described in the subsection 4.1.2 is highly theoretical, which makes it challenging to evaluate. Therefore, the method-artifact was used to create a minimum-viable-product implementation of one possible instantiation of it. This implementation is used during the case study at Visma, and the data generated about the implementation can be generalized and used to evaluate the method-artifact. To ensure that the date can be transferred in this way, the implementation must share at least some requirement with the method artifact. The implementation took quite long time, as it needed to be approved by the security team

at Visma, before it could be put into action with a development team. Therefore, two security experts from Visma participated in weekly meetings during the development process, to help guide the development and ensure the quality of the implementation before the case study.
The implementation is available as a GitHub action on the GitHub marketplace `https://github.com/marketplace/actions/pr-question-template-action`.

The development began by creating a list based on the requirements presented in subsection 3.2.1, and then determine which elements needed to be developed. This list is shown below:

1. Activate the artifact when a pull request is created
2. Gather code changes (two options)
    - Automatic
      or
    - Manual
        a. Ask questions
        b. Block PR from being merged if the questions are unanswered
        c. Gather answers
3. Calculate PR contribution index
4. Store results in a database
5. Calculate accumulated contribution index
6. Check if the accumulated contribution index has surpassed the threshold
7. If a trigger: notify the team

This implementation were challenging as no similar systems were found, but it is still a functional minimum-viable-product implementation of the method-artifact. It was quickly decided that the automatic calculation of contribution index would not be implemented, and therefore not the triggering part as well. The same goes for the automatic detection of types of code changes. Instead, a GitHub action was created, that gathers answers from the developers, and then stores them in a database with a reference to that PR. Then, when the caste study was over, manually calculations could be performed on the raw results. This makes it possible to perform the necessary evaluations for the case study, to evaluate the method-artifact.

The implementation is quickly summarized below, and then further explained and discussed. The developers create pull requests as they normally do, but now they must answer a set of questions, like the ones presented in Table 4.2, inside their pull request. The GitHub workflow detects the new pull request and executes the GitHub action. The GitHub action extracts the answer from the pull request body and creates SQL files that will update the database. The GitHub workflow then executes these files. This functionality is illustrated in the the sequence diagrams shown in Figure 4.3 and the GitHub action is illustrated in Figure 4.4.
This implementation uses three principles: pull request template[1], GitHub workflow[2], and GitHub action-footnotehttps://github.com/features/actions. An Azure SQL database was used to store the data. Each of

[1]https://docs.github.com/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/creating-a-pull-request-template-for-your-repository
[2]https://docs.github.com/en/actions/using-workflows

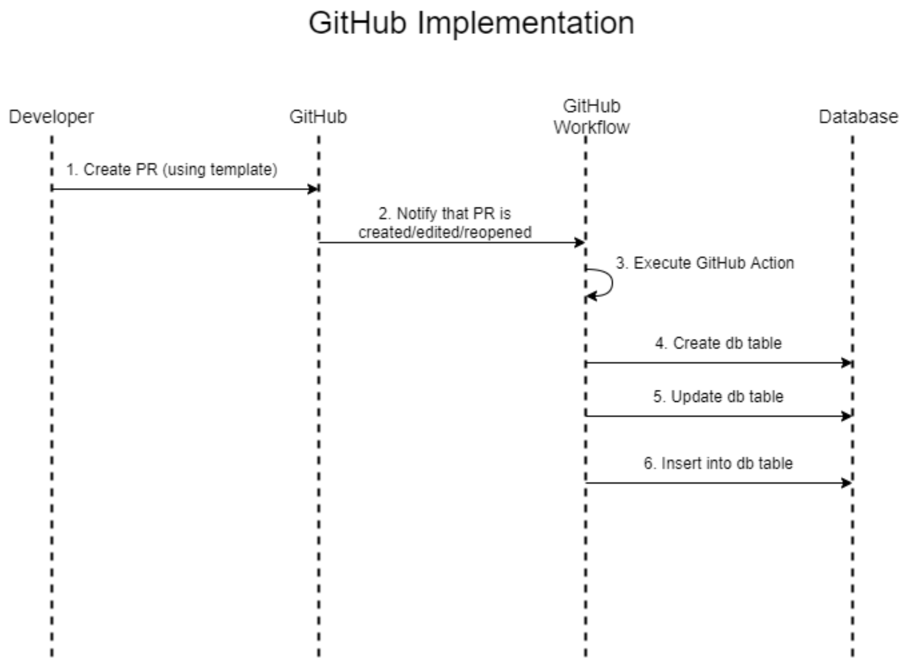


Figure 4.3: A sequence diagram illustrating the GitHub workflow

these will be explained in detail below.

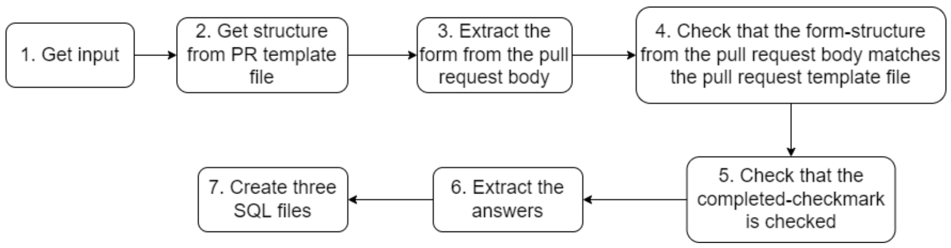

1. Get input
2. Get structure from PR template file
3. Extract the form from the pull request body
4. Check that the form-structure from the pull request body matches the pull request template file
5. Check that the completed-checkmark is checked
6. Extract the answers
7. Create three SQL files

Figure 4.4: An illustration of the steps of the GitHub action

The **sql database** consists of multiple columns. The first is the id field. The ID field needs to be unique, and it should therefore be something than can identify the pull requests uniquely. The SHA hash of the pull request fulfills these requirements, and was therefore used as ID. This gives one major advantage, since IDs needs to be unique, then one pull request will never be stored as multiple rows, as the pr SHA doesn't change even if you add more commits. But, this means that the collision of IDs needs to be handled. The next rows are the answers to the questions, and the final row is a date-stamp. This structured is illustrated in Table 4.3.

The **pull request template** is a way of automatically inserting content into the body of a pull request. I

Table 4.3: An illustration of the structure of the database schema used in the case study

PR SHA hash	Answer 1	Answer 2	...	Answer 20	Date Stamp

used this principle to insert the form with the questions into the pull requests when they are created. This file needs to be stored inside the `.github` folder inside a GitHub repository. The template that was used can be found here `https://github.com/KristofferHaakonsen/pr-question-template-action/blob/releases/v2/.github/pull_request_template.md`, except for that the "none of the above"-option was not present during the case study, and the explanation at the top of the file was therefore also a bit different. This file is automatically inserted as the body of a pull request. Different approaches was tested during implementation, the most promising was having a bot inserting the questions as a comment, but this was not possible to implement within the time constraints. The PR template file also contains some tags "HTML" < >, which is not possible to see without viewing the raw file. These are used by the GitHub action to locate the form. Because it is very likely that the developers will write some descriptions as part of the pull request body, and the form must therefore be extracted from the rest of the descriptions. A checkbox is placed at the bottom of the template, which the developers must check when they have filled in the form. This checkbox is used to inform the GitHub action that the form is completed, as a form can be completed even if it does not contain any answers. This is because an answer can be left blank if the answer is "none" or "0".

GitHub workflows are automatic actions that GitHub can execute based on different triggers. The workflow used in the case study can be seen in the following file in GitHub: `https://github.com/KristofferHaakonsen/pr-question-template-action/blob/releases/v2/.github/workflows/test.yml`. It is triggered if a pull request is opened, edited, or reopened. It begins by calling the GitHub action, which will be explained in the next paragraph. The GitHub action extracts the questions and creates sql files for the database actions. Then, the workflow executes the sql files using the azure/sql-action[3], which makes database operations simpler. The functionality of the GitHub workflow is illustrated as a sequence diagram in Figure 4.3.

GitHub actions are programs that can be executed from a GitHub workflow. The action is written in JavaScript, and v2 is the version that was used for the case study `https://github.com/marketplace/actions/pr-question-template-action`. Figure 4.4 illustrates the steps the GitHub action performs.
The GitHub action parses the body of the pull request and extracts the form, using the "HTML" tags as described in the paragraph about the template file. Then it checks that the checkbox at the bottom of the form is checked, if it isn't, then this action fails and the workflow returns an error. If it is checked, then it checks that the structure of the form is correct, by comparing the form from the pull request body, with the form in the template file. It then extracts the answers, a number, from each questions. If there is no answer, it is set to 0. Then it creates three sql files. The first is to create a database table, the second is for inserting the data as a new row, and the third is to update the existing row with the new data from the pull request. This might seem weird, but it has a simple explanation. The azure/sql-action does not provide good feedback to the rest of the workflow. So, it is not possible to determine how the database responded to the execution of the sql code, as it always returns the status code 0. The implementation prepares for the

[3]https://github.com/marketplace/actions/azure-sql-deploy

worst, and assume that the database table is not there. But this is a feature that should be able to toggle, to reduce the load on the db. Other alternantives were concidered instead of the azure/sql-action, but no other similar actions existed. Several attempts was made to create a similar database action for this case study, but this took valuable time away from the research, and it was abandoned after discussions with the security experts. It was simply better to use the azure/sql-action, even though it did not provide the output the implementation needed.

Then, the implementation need to insert the data into the database. But, it cannot know if this pull request already has an entry in the database, because an insert statement will give status code 0, even if it is not inserted due to ID collision. In MySQL[4] it is possible to perform an insert statement that is executed as an update statement if there is a collision on the ID field [36], this is not possible in SQL. Therefore, the implementation also performs an update action, in case this PR already exists in the database. It executes the update statement first and then the insert statement, as this means that sometimes only one sql operation is executed in the database, as compared to required two if the update were done after the insert. As the update statement always will modify the table if it is done after the insert. One could also have first gotten the existing IDs in the database and then performed either an insert or an update, but this offers little performance boost but a lot more code.
These steps are illustrated in Figure 4.4.
The implementation fulfills functional requirements 1-3, except for the calculation and storage of the pr contribution index. But, instead it stores the answers. This makes it possible to mimic the functionality of functional requirements 4-6. FR7 is partially implemented as the questions can be changed by editing the pull request template file, but the contribution factors and threshold cannot be changed as they are not implemented. It is challenging to determine how well this implementation fulfills the non-functional requirements, before the case study is performed. But answering 6-10 questions whenever a developer creates a pull request is not very much, and one can therefore argue that NFR1 is at least partially fulfilled. NFR2 is harder to evaluate. NFR3 is fulfilled as all interactions take place within GitHub. NFR4 is dependant on the questions that the developers will answer.
All in all, this implementation resembles the method artifact, as it fulfils or mimics most of the functional and non-functional requirements. Therefore, data generated about this implementation can be used to evaluate the method-artifact.

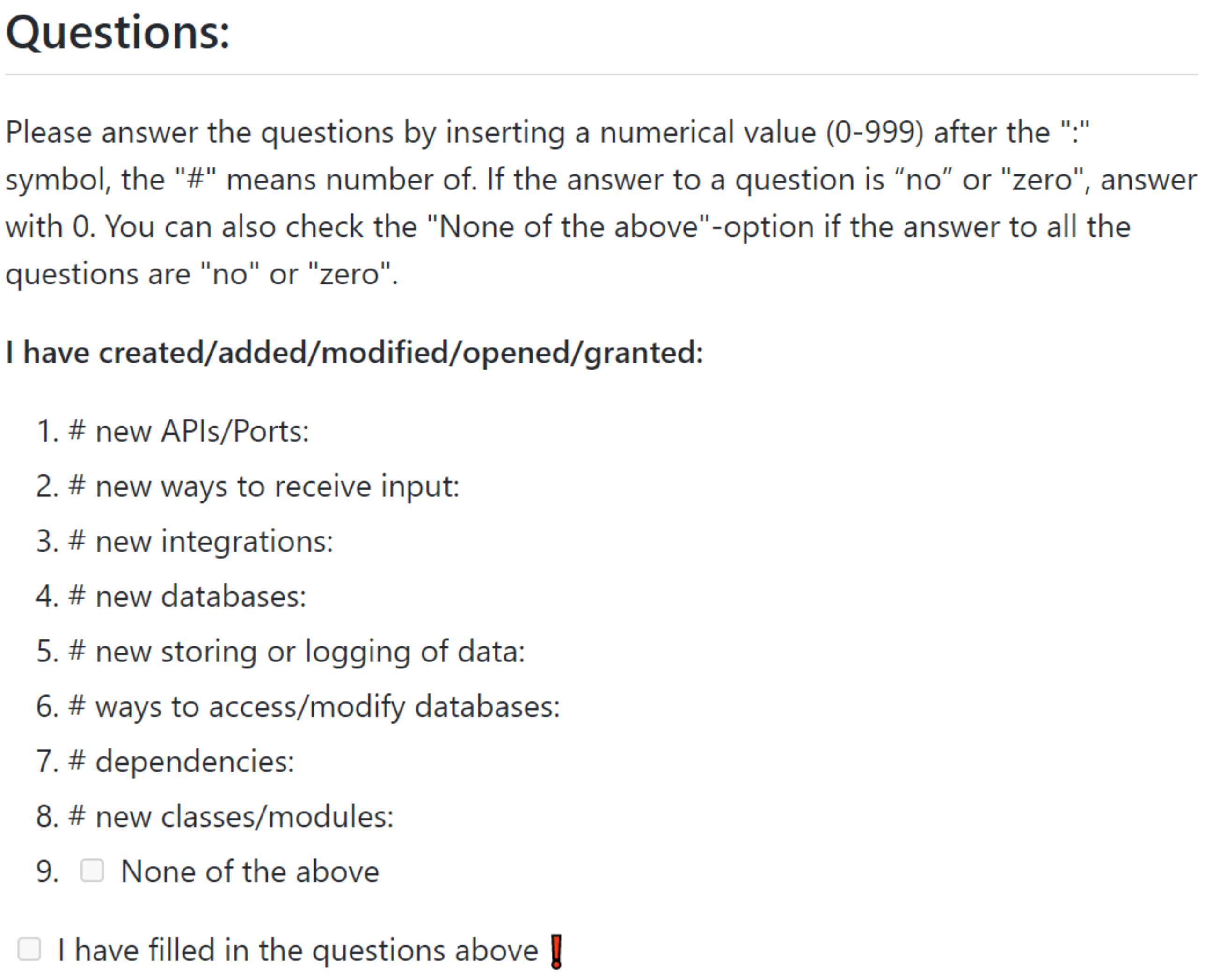
More information about the GitHub action and workflow can be found in the marketplace page of the action: `https://github.com/marketplace/actions/pr-question-template-action`. The raw output generated by this implementation presented in the next section.

4.4 Case study raw data

This section contains the raw data generated over a period of about three weeks, 18 workdays, by a feature team at Visma, using the implemented GitHub action and Workflow as described in section 4.3. The questions that the developers needed to answer were determined in advanced. Multiple different questions and form-layouts were discussed with security experts, before they were tested in the paper prototype tests.

[4]https://www.mysql.com/

They were then modified based on the results and final discussions with security experts at Visma. The question and form-layout that was used during the case study is shown in Figure 4.5.



Questions:

Please answer the questions by inserting a numerical value (0-999) after the ":" symbol, the "#" means number of. If the answer to a question is "no" or "zero", answer with 0. You can also check the "None of the above"-option if the answer to all the questions are "no" or "zero".

I have created/added/modified/opened/granted:

1. # new APIs/Ports:
2. # new ways to receive input:
3. # new integrations:
4. # new databases:
5. # new storing or logging of data:
6. # ways to access/modify databases:
7. # dependencies:
8. # new classes/modules:
9. ☐ None of the above

☐ I have filled in the questions above ❗

Figure 4.5: Screenshot from GitHub of the form containing the questions that was used during the case study

During the case study, the developers answered the questions whenever they created the pull requests, and the results were stored in a database. This data is anonymized and included in Appendix D, the dates and commit hashes has been replaced with numbers. The raw output from the artifact is presented in as a graph in Figure 4.6. From this figure we can see the answers to the eight questions, plotted against the 18 workdays the case study lasted. We can see that questions 2,3, and 4 never got any answers. Question 8 got the most answers, which one would expect, as it represents new classes and methods. Each question is illustrated in Figure 4.5. Just seeing the number of answers to each questions for each day does not bring that much new insights, but it is possible to perform some simple analysis on this data.

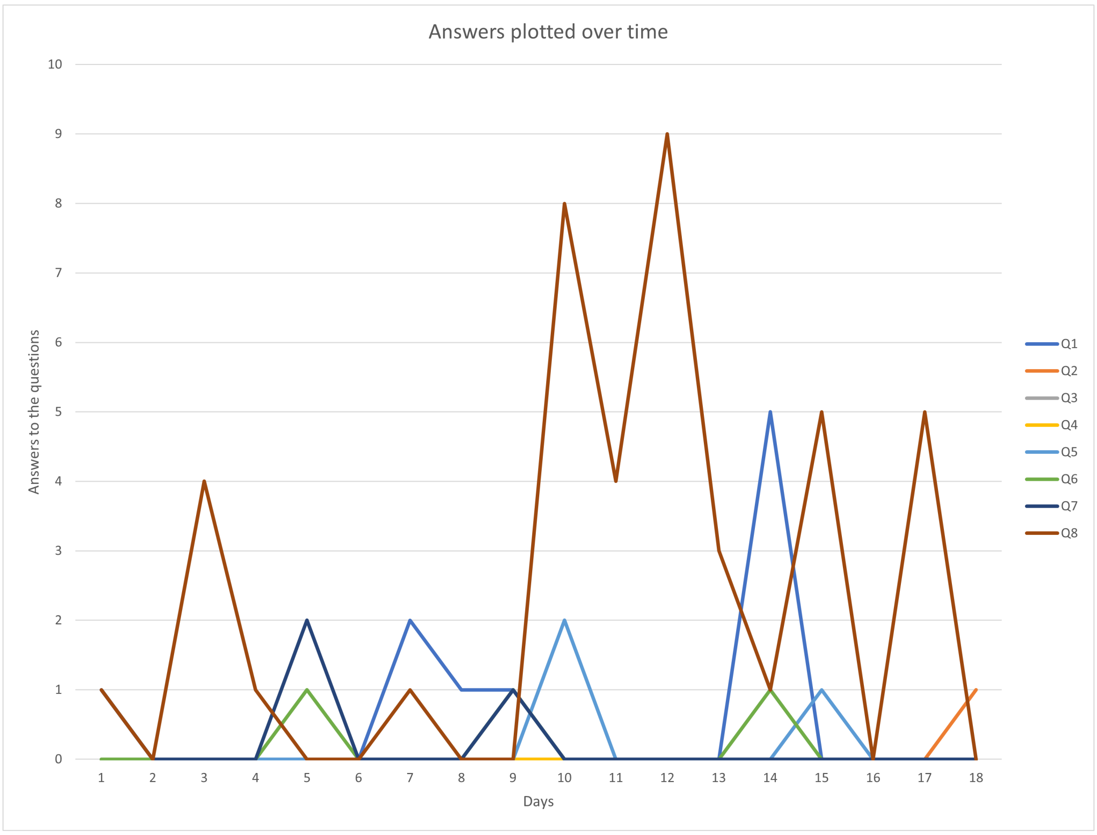


Figure 4.6: The number of answers for each questions plotted against days into the case study

Figure 4.7 shows the same data, but this time, the answers are accumulated to show the total number of answers for each question at each day. This is the same as the contribution index from each question, without contribution factors. These contribution indexes have been summarized to calculate the accumulated contribution index, without contribution factors, as seen in Figure 4.7. This graph clearly shows the different changes that is introduced to the codebase by the developers, which is the fundamental principle of this artifact. This accumulated contribution index could be used to trigger threat modeling itself, but it is probable that it is less precise than if the contribution factors were added. However, the contribution factors will need to be adapted and maintained over time to be well adapted to each team and their code base.

Figure 4.8 shows the same data as Figure 4.7, but this time, contribution factors has been added. The contribution factors have been chosen by the researcher and is added to show how these factors will impact the results, and are based on my own best recommendations. The contribution factors are presented in Appendix D. Figure 4.8 shows how some questions have a larger impact on the contribution indexes, compared to other. This can clearly be seen as the contribution index for question 8 is nearer the other indexes in this graph, compared to the ones presented in Figure 4.7. This is natural as question 8 is about new classes and methods, which is a common type of code change, which therefore should have a low contribution factor, compared to other types of changes. This graph shows the more advanced calculations that this artifact can do, and the accumulated contribution index could be further calculated with more

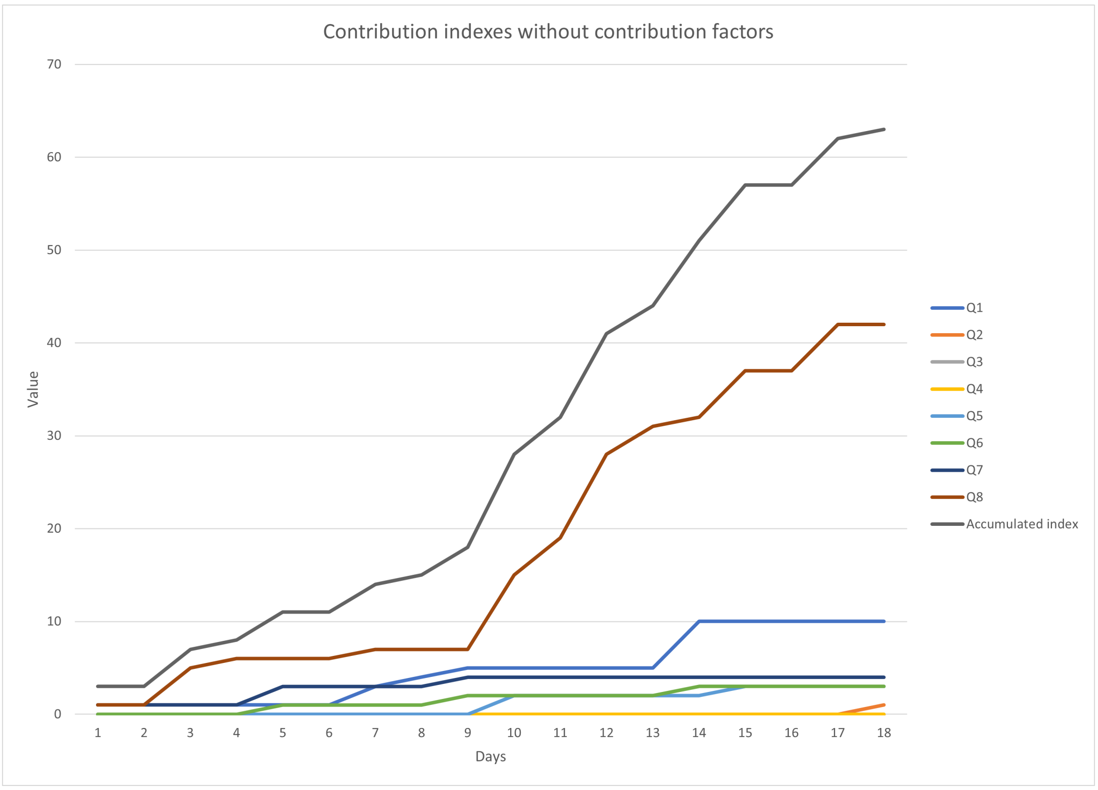


Figure 4.7: The contribution indexes for each question and the accumulated contribution index plotted against days into the case study, without contribution factors.

advanced formulas.

If we imagine that this team had defined a threshold for threat modeling when the accumulated contribution index reached 320, then we can see from Figure 4.8 that they were about one day away from a trigger for threat modeling.

Another way of presenting these data is to just view the total number of answers for each question, as presented in Table 4.4. This provides a quick overview of the total number of change within each type that the team has performed within the last 18 days.

The data presented in these graphs and in the Appendix D are quite similar to the ones presented in the Appendix A, and it could indicate that the artifact is working as intended. The usefulness and other evaluations performed on these data is presented in the section 5.3.

The complete data is presented in Appendix D, and the results of the evaluation is presented and discussed in section 5.3.

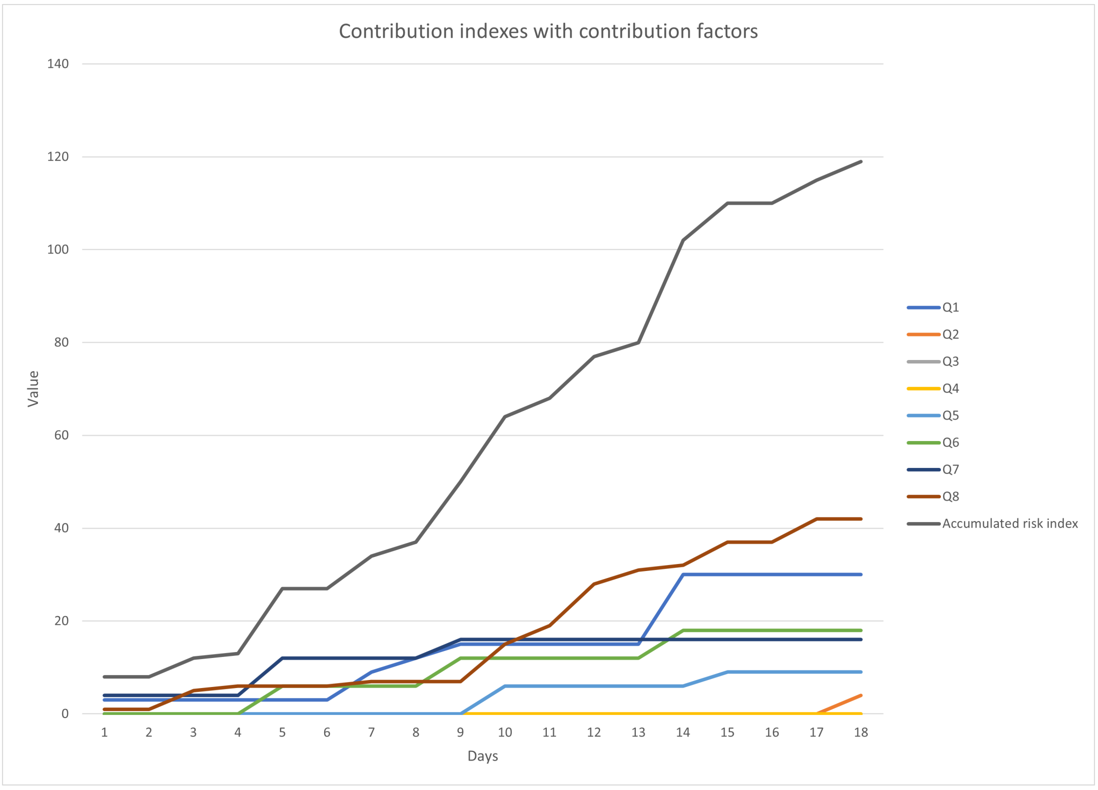


Figure 4.8: The contribution indexes for each question and the accumulated contribution index plotted against days into the case study, with contribution factors.

Table 4.4: The total number of changes for each questions gathered during the case study

# new API/-ports	# new ways to receive input	# new integrations	# new databases	# new storing or logging of data	# ways to access/ modify database	# dependencies	# new classes/ modules
10	1	0	0	3	3	4	43

Chapter 5

Evaluation

The fifth step of the method framework, as described in subsection 3.1.2, is to evaluate the artifact. This chapter begins by describing the practical data generation, before describing the evaluation goals, and then finally presenting the evaluation results.

5.1 Data generation

The data is generated in three different ways: two user tests with SpareBank 1 Utvikling, one case study at Visma, and an informed arguments, as described in section 3.3
The first user test was used as a formative evaluation during the first iteration of the design. The second user test and the case study at Visma were done during the second iteration. This means that the first user test is not directly comparable to the second user test and the case study, as they have tested slightly different artifacts. At the same time, the designs were similar, and some comparison is therefore possible.

5.1.1 User tests

A paper prototype of the artifact were tested with two different development teams at SpareBank 1 Utvikling. The first test was done during the first iteration of the design, and the results were used as a formative evaluation to improve and guide the design. The first test were done physical with six developers, while the second were done digital with 4 developers. The paper prototype was a piece of paper with the questions shown in Figure C.1. This method artifact should ideally be implemented with automation, to such an extent that the developers will not need to interact with it other than when they get notification. This paper prototype tests the opposite, a situation where the system is not automated, and the developers must tell the artifacts about their code changes. This means that the results of these tests represents the worst-case implementation of such an artifact. These results should therefore be interpreted as such.

5.1.2 Case study at Visma with GitHub action

To be able to evaluate if the chosen questions can represent the code changes and be accumulated into a contribution index, data from pull requests are needed. This information can also be used to evaluate if the data about the types of changes could be useful by itself. A team of developers at Visma have therefore been documenting their code changes over a period of 3 weeks, or 18 work days, by using the implemented artifact described in section 4.3. This closely resembles the interaction the developers will have in a non-automated implementation of the artifact described in subsection 4.1.2. Therefore, this testing generates both data about the actual code changes within some different categories represented by questions, the usefulness of the this data, and also experiences with such an approach over longer periods of time. The case study ended with an interview with all the participants from the case study. In addition, one interview was perform with the head of security development and a senior infrastructure engineer. This last interview were a semi-structured interview, where they studied the data generated from the artifact itself over the three weeks, and also some simple analysis of the data. The data that was presented in shown in section 4.4

5.2 Evaluation descriptions

The tasks in step five of the design science framework are to select the evaluation goals and strategy, before evaluating the artifact [18].
There are six goals in the evaluation step of the design science research framework [18]. The evaluation contains five of these and an investigation of the usability of the artifact. Each questions and metric in the data generation is tied to one or more of these. The six goals are presented in Table 5.1.

Table 5.1: The evaluation goals linked to data generation

Evaluation goal	Data generation
1. Evaluate to what extent the artifact is effective at solving the problem.	Questions 10, 14, and 15. Ex ante argument.
2. Investigate the usability.	Interview questions 19. and 20. SUS form. Timer.
3. Evaluate the artifact against its requirements.	Interview questions: 8, 9, 11, 12, and 13. How long time the participants spent during user tests. Ex ante argument
4. Compare the artifact to other similar artifacts.	Interview question 19.
5. Investigate side effects of the artifact.	Interview question 16.
6. Perform formative evaluation to improve the artifact.	Interview question 5, 6, and 7. And general feedback from participants.

5.2.1 To what extent is the artifact effective for solving the problem?

To determine how effective the artifact is at solving the problem, we must be clear about exactly what the artifact does. It accumulates the contribution index and notifies the teams. As mentioned before, the part where the system notifies the team is not interesting to test, as it could be done with a slack bot or an email. Instead, the tests should be focused on how effective the artifact calculates the contribution indexes. In other words, how well does the artifact classify the code changes, how well does these classifications represent the different types of code changes in the system, and how correct are the contribution factors.
It is important to note that we cannot test the types of code changes directly, as the artifact implemented in this thesis relies on questions, which the developers need to answer. But, we can evaluate how good these questions are, at detecting potential risky code changes and how well they are adapted. But, these questions and their contribution factors should be adapted to each team, and the same with the contribution factors, which means that any evaluation we perform will not capture the evolution and adaptation of those.
This will therefore, be evaluated through some questions in the user tests, the case study, but also one final semi-structure interview with the leader of the security team and a security expert from Visma.

5.2.2 Usability

Usability was measured with both interview question, timing of the user tasks, and also through a system usability score form (SUS). The SUS measures usability directly, while the other metrics measure either effectiveness, efficiency, or satisfaction, as described in section 3.3.

5.2.3 Evaluate requirements

The implemented artifact should be evaluated against the requirements defined in subsection 3.2.1, to evaluate to what extent it fulfils them. It is difficult to measure how well this artifact fulfils the functional requirement, as the artifact itself is a method. The evaluation of the functional artifacts can therefore be done with an *informed argument* [18].
The non-functional requirements is easier to measure. Table 5.2 shows how the different NFRs can be linked to interview questions or other data generated in this thesis. The evaluation is based on data from the two paper prototyping user tests, the two interviews from the case study at Visma, and an *informed argument*.

Table 5.2: Non-functional requirements for the artifact linked to their evaluation

NFR ID	Evaluation
NFR1	Interview questions: 11, 12, 13, the time it took the participants to answer the questions
NFR2	Interview question: 8
NFR3	Informed argument
NFR4	Interview question: 9

5.2.4 Compare artifacts to similar artifacts

This artifact is compared to similar artifacts in two ways. The participants are asked how they perceive the artifact compared to other similar artifacts. The internet and other publications are checked for similar artifacts, which is compared to this artifact.

5.2.5 Investigate side-effects

During the early phases of this project, the artifact was discussed with security experts from both SpareBank 1 Utvikling and Visma, and among other aspects, potential side-effects were discussed. The only potential side effect mentioned were a change in the size of the pull requests. No side-effects were suggested if the artifact was automated. If it is not automated, the only side-effect that came up was a potential change in the culture around pull request. The developers could for example create larger PRs, to reduce the number of times they need to answer the questions, or they could be smaller, so that the questions are easier to answer. One questions were therefore added to both the user tests and the case study interviews, to try to capture any information about this potential side effects. In addition, the participants might mention something during the more open-ended questions.

5.2.6 Formative evaluation

A formative evaluation is when an artifact is evaluated with the goal of improving the artifact, in comparison to summative evaluation, where the goal is to assess the artifact [18]. All open ended questions in both the user tests and the case study could contribute to the formative evaluation, as well as interview questions 5,6, and 7, and the observation during the user tests.

5.3 Evaluation results

This section presents the results from the data generation and analysis, grouped together by the goals of the evaluation. The results are presented and then lightly discussed. A more detailed discussion is found in the chapter 6

5.3.1 Participants demographics

The first user tests were performed with members from a team of developers from SpareBank 1 Utvikling. They were six developers working in a fullstack team, 2 mainly worked with front end, 2 mainly with back end and 2 mainly fullstack. The second user tests were performed with four members from a mainly back end team at SpareBank 1 Utvikling. Most developers from both teams had worked as a developer or tester for more than 5 years, and many had worked for more than 10 years. The case study at Visma were done with a feature team, with five developers, one tester, and two security experts. Most participants had worked as a developer or tester for more than 5 years. The developers from both companies reported that they prioritize security over new features, and most agreed that their security skills were average, while a few rated themselves as highly skilled in software security. The full meta data is presented in Appendix C.

5.3.2 To what extent is the artifact effective for solving the problem?

The participants of both user tests and the case study were asked several questions that could be used to evaluate how effective the artifact is at solving the problem. For the user tests, these questions were about a theoretical description of the fully implemented artifact, as the paper prototype itself is harder to evaluate. The theoretical description can be found in section C.2. During the case study, the participants answered the same questions for both the implementation they had tried, as well as for the same theoretical description as in the user tests.

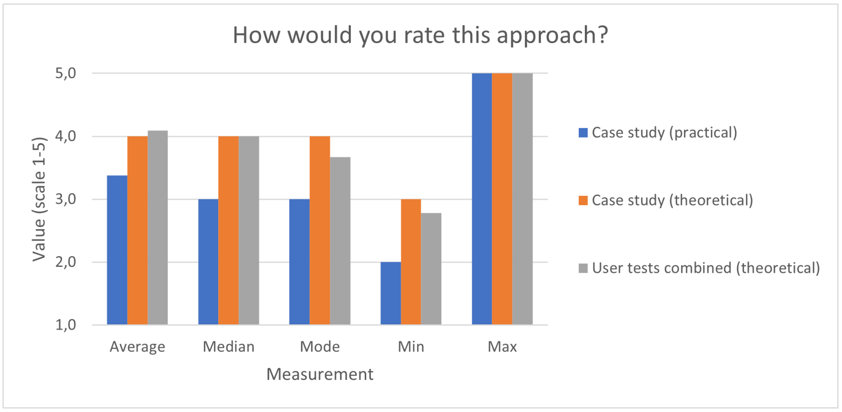


Figure 5.1: How the participants would rate this artifact on a scale from 1 to 5. The data is from both the case study and the normalized data from the two paper prototypes, for both the theoretically described artifact and the implemented one used in the case study.

The participants were first asked to rate the artifact on a scale from 1 to 5, then they were asked how useful they think this artifact would be, and finally, if it would be worth the extra hassle/motivation. The results for both the theoretical description and the implementation tested in the case study is shown in Figure 5.1, Figure 5.2, and Figure 5.3. Both the rating and the usefulness of the artifact is rates as above average for both theoretical and practical implementation. We can also see that the theoretical artifact got a higher score in each measurement. This was expected, as a more automated and mature artifact sounds better than a simple prototype. It is clear that the developers think that this artifact is very useful.

The final interview with the head of security development and a senior infrastructure engineer provided interesting data and viewpoints to the results from the case study, presented in section 4.4. Most of the interview contained open-ended questions with answers in the list below, but also one quantitative question. When asked to rate the usefulness of the data produced by this artifact, they rated it a solid 5, on a scale from 1-5, they even added that it would be a 10 if they scale had gone that far. The interview can be summarized as: they see a huge potential for this artifact and they believe that the output is very useful. They also revealed that they did not take down the instantiation, but they continue to use it as part of their pipeline to generate data. More specific results are shown in the list below:

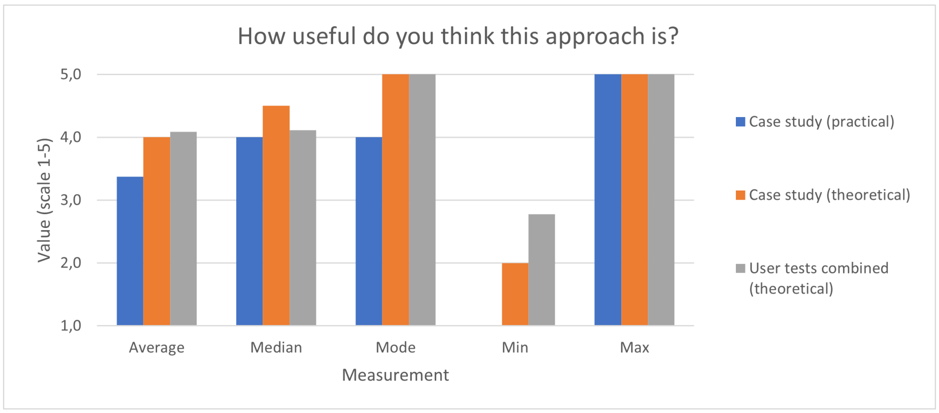


Figure 5.2: How useful the participants think the artifact was. The data is from both the case study and the normalized data from the two paper prototypes. The data is from both the case study and the normalized data from the two paper prototypes, for both the theoretically described artifact and the implemented one used in the case study.

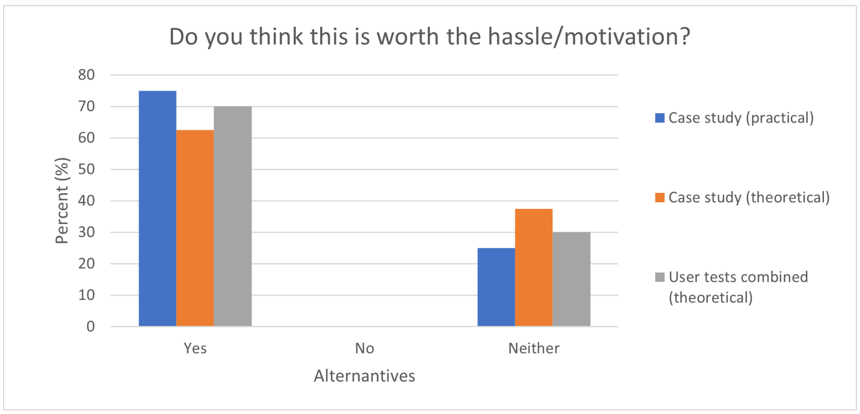


Figure 5.3: How the participants would rate this artifact. The data is from both the case study and the normalized data from the two paper prototypes. The data is from both the case study and the normalized data from the two paper prototypes, for both the theoretically described artifact and the implemented one used in the case study.

- I find the new insights very useful.
- They wondered if the artifact could be extended with multiple different tables, so that for example front end and back end contributed to different threshold.
- They suggested an additional contribution factor for each question, that indicated the security importance of the code.
- I think it is useful for multiple reasons: The first is the accumulated contribution index graph that will trigger the threat analysis, but I also think that we can find some patterns on how the team works, based on the questions, and improve the way the team works based on the patterns.
- When studying the raw data, without the contribution factors, we can use this to estimate the velocity over time. As we can see how much the team did in one quarter, and then we can see next quarter if they have a high number of changes here. And then do some analysis. And measure a bit what they

are working on, just by this data. So not only looking at tickets and story points, but at the actual security concerns and the complexity that has been introduced over time.
- I think showing the raw data as a dashboard would be a very good tool when you review your SSA, and I also see many other uses for this data.
- I love this idea; I think it is really great! I think it is really amazing.
- I think this idea is super awesome, and I was very thrilled to work on it. It is multiplied stuff we can continue to implement and continue to work on.

As seen from the list above, both the participants see huge potential for this artifact, and think the data, both raw and analyzed, is highly useful. This raw data consists of a table of pull requests and the number of changes within each type of change, and the pr contribution index. Both participants believed that this data was valuable itself. This opens up possibilities for other uses than just the triggering, such as dash boarding. In addition, it creates confidence to know that a team has not performed any of a certain type of code change, such as creating a new database. This data could for example be reviewed at the end of a sprint to check how many and what type of code changes they have made. So the raw data has value, even if it is a 0 in a type of code change, as it gives confidence that it has not been done. It is interesting to see that both the developers and security experts think this artifact is useful. This brings more confidence to the method-artifact and indicates the need for further research.

This artifact has was evaluated as effective for solving the problem, and the results are highly useful for security experts and developers. The second interview after the case study revealed that the output had multiple different uses, such as dashboards, data for sprint reviews, team analysis and more. We can clearly state that this artifact is useful for solving the problem, and it is likely that a fully automated version could be even more useful.

5.3.3 Usability testing

The system usability score (SUS) from both user tests, case study, and the combined results is shown as average, median and mode, in Table 5.3. The average SUS for the paper prototyping was 83, and the average for the case study was 75. This result is higher than the average result of a SUS form, which was 68, but ideally, the score should be above 80 [28]. But all in all, the results are high, within the acceptable range, and above average. We can therefore infer that the usability of the non-automated version is at least acceptable, and one can assume that the automated version will score even higher.
It is also interesting to note that the paper prototype received a higher score than the implementation, this could be for a number of reasons. One possible reason could be that the developers tested the paper prototype once, while the case study were tested over a period of three weeks. In addition, the implemented version in the case study are subject to practical challenges that is impossible to capture within a paper prototype user test.
All details regarding the SUS scores and the calculations can be found in Appendix C.

The participants were asked some open-ended questions, at both the user tests, and at the interview after the user tests, the results related to usability is presented in the list below:

Table 5.3: The SUS scores from the two user tests, combined sus score for both user tests, the case study, and the combined SUS score. It is indicated with average, median, and mode, rounded to nearest whole number.

Source	Average SUS	Median SUS	Mode SUS
Paper prototype 1	85	83	80
Paper prototype 2	80	81	83
Combined paper prototype	83	83	83
Case study	75	76	78
Combined	79	80	83

- Most participants understood all the questions, but there were some uncertainties regarding the questions, mostly about what was meant by specific questions.
- Several participants of the paper prototype tests commented that the number of questions needed to be as few as possible.
- Some of the questions feels unrelated for front end development.
- Some developers pointed out that there would be necessary to inform future developers about why they need to answer these questions, as they might not do it if they doesn't understand why.
- Most participants commented that this has a huge potential for automation, and that it feels pointless to be forced to answer something that should have been auto-detected.
- After the case study, the participants told that if they didn't complete the form immediately when creating a PR, then it failed the pipeline, which sent email to code owners and the developers, which was annoying.
- During the case study, one tester also tested the artifact indirectly as part of his/her work. This was reportedly not useful, as he/she never answered the questions, but used PRs for tests and similar activities.
- If a developer answered the questions, but then later added a new commit to the PR, the questions didn't reset.

It is clear from the feedback in the list above, that the artifact has room for improvements when it comes to is usability. Several of these challenges can be mitigated by automation, such as reducing the number of questions and that it feels pointless to answering something that should be automated. Challenges related to different questions for different parts of the code to ensure that they are relevant, will not be mitigated by automation unless it is fully automated so that the developers will not need to interact with the artifact at all. Specific questions for front end is an example of this type of challenge. The uncertainties regarding specific questions will hopefully be improved if a team adopts this artifact and spends some time on maintaining the questions and improving them, and the same goes for the technology acceptance. Finally, the quick failing of the pipeline and spamming email is also something that drastically impacts the usability, but this is not a problem with the method artifact itself, but instead on the implementation of the artifact. The same goes for the answers not resetting when a new commit is added. Multiple of these challenges are related to the instantiation and not the method-artifact, which is positive, as these could be mitigated with tweaks to the instantiation.

The efficiency of the interaction with the artifact is measured by asking the participants to estimate how long time they averagely spent answering the questions inside GitHub. The average time was 38 seconds, and can be seen in Table 5.4. 38 seconds is quite fast, especially since they tested a completely manual version of this artifact. This time should ideally be reduced, which could be achieved by automation.

Table 5.4: The results when the eight participants were asked how long time they on average spent answering the questions inside GitHub.

Measurement	Time (seconds)
Average	38
Median	27
Mode	60

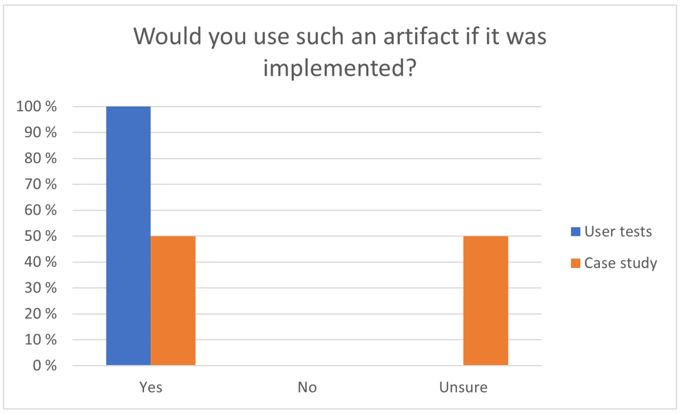


Figure 5.4: The participants of the second paper prototype user test and the case study were asked if they would you use such the theoretically described artifact if it was implemented?

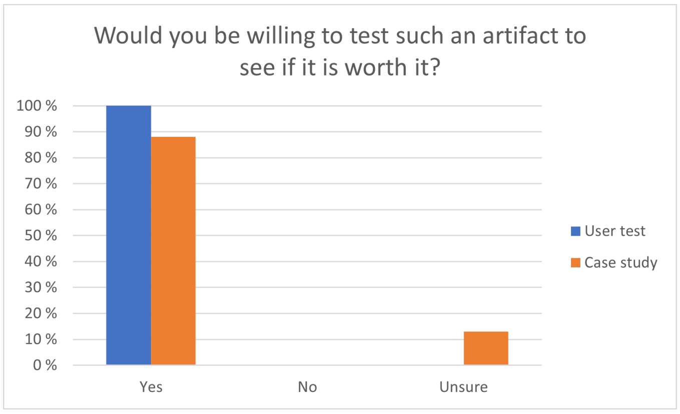


Figure 5.5: The participants of the second paper prototype user test and the case study were asked if they would be willing to test the theoretically described artifact if it was implemented to see if it is worth it?

Finally, all four participants from the second user test and half of the participants from the case study said that they would use an artifact like the theoretically described one, if it was implemented, as indicated in Figure 5.4. In addition, all four participants from the second user tests, and all but one participate from the case study would be willing to test the theoretically described artifact if it existed. This clearly shows that the participants see the usefulness of this type of artifact. We notice that the participants in the case study were less motivated to use such an artifact, this is probably due to the annoyances in the GitHub implementation that they used. This artifact is primarily a tool for the security experts and managers, but developers must insert data into it in the manual form. It is a clear indication that the developers see its use, when they are as willing to both try it and use it.

All in all, the usability of the artifact is good, within the acceptable range. It has room for improvements, but this is expected from an such a newly defined artifact.

5.3.4 Evaluate requirements

Functional To evaluate how well the artifact fulfills the functional requirements, presented in subsection 3.2.1, I will present an *informed argument* as described by Johannesson and Perjons in [18]. The following is my own informed argument:

The artifact contains a list of types of code changes, that can be represented by questions with related contribution factors. The developers needs to answer these questions whenever they create a PR, and the results are stored in a database with reference to the PR. The contribution index for each PR is calculated and summarized leading to the accumulated contribution index. When this index reaches a threshold, the team will be notified and they should perform threat modeling. If they do that, they can reset the accumulated contribution index, to "start from the beginning". The artifact is flexible and the types of code changes, questions, index factors, and threshold is modifiable. I will argue that this fulfills functional requirements 1-7.

Non-functional Non-functional requirement 1 can be evaluated by studying the results from four of the interview questions, presented in Figure 5.6, Figure 5.7, Figure 5.8, and Figure 5.9. The participants in the interview at the end of the case study were asked questions both about the implementation they have tested (practical), and about the theoretical artifact, while the participants in the paper prototyping user tests were only ask about the theoretical artifact. It is clear from Figure 5.6 that the implementation impacted the development teams workflow very little, and we can see that it slowed there development down very little, from Figure 5.7. The participants reported that the artifact required very little extra hassle/motivation from them, as seen in Figure 5.8. This also indicates that it impacted their workflow very little.

On average, the participants in the case study reported that they spent less than 40 seconds on answering the questions. We can also see that it impacted their workflow to a small degree, from the theoretical questions, both from the case study and the paper prototyping user tests. These results indicate that non-functional requirement 1 is fulfilled, as the artifact impacted the agile development to a small degree. It is however interesting to note that some participants in the case study reported that this artifact impacted and slowed down their work a lot, as we can see that at least one participant has answered 4 and 5. This

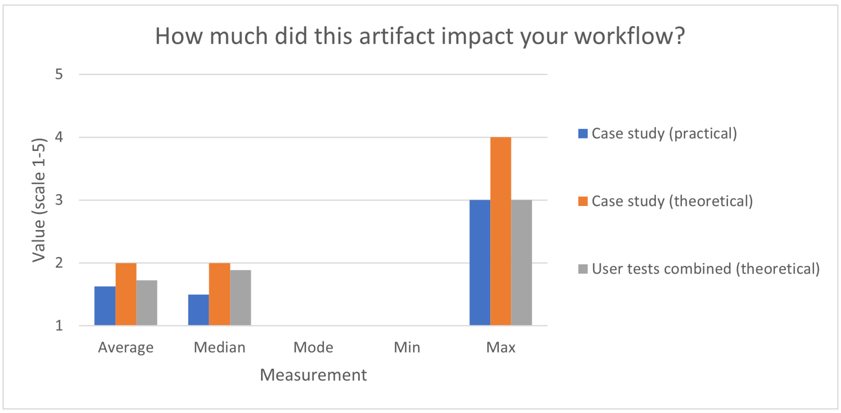


Figure 5.6: How much the participants think this artifact impacts their workflow, on a scale from 1 to 5. The data is from both the case study and the normalized data from the two paper prototypes. The data is about the theoretically described artifact and the implemented one used in the case study.

is very interesting and could be for many different reasons. One possible reason is that this artifact has impacted non-developers workflow a lot, this was also reported on one of the more open ended questions: "*for non developers is it a tiny minor inconvenience, as we will never answer the questions*".

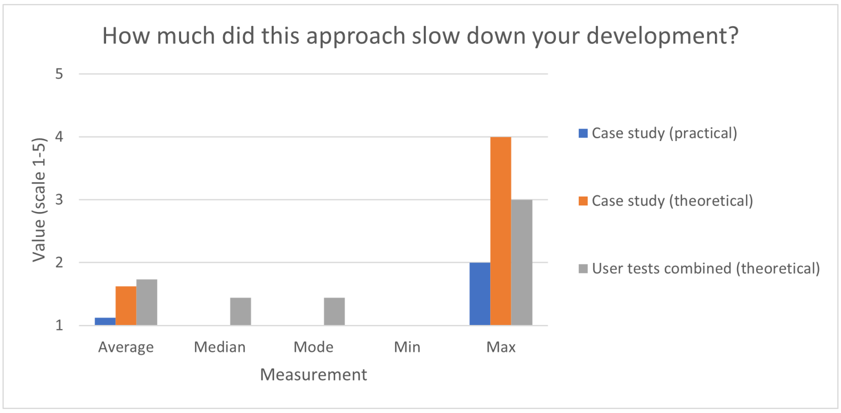


Figure 5.7: How much the participants think this artifact slows down their workflow, on a scale from 1 to 5. The data is from both the case study and the normalized data from the two paper prototypes. The data is about the theoretically described artifact and the implemented one used in the case study.

NFR2 is harder to evaluate as it there are no academic sources that has studied what types of code changes that can be used for triggering threat modeling, as previously discussed in the section 2.4. But, half of the types of code changes are inspired or identical to the ones presented by SAFECode [16], and these questions therefore have support from other publications than just this master thesis. In addition, the participants were asked how relevant the questions were for the code base they worked on, the results is shown in Figure 5.10. It is clear that many of the participants in the paper prototype user tests found the questions relevant, as the average answer was about 4, on a scale from 1-5. However, it is also clear that the development team that participated in the case study did not find the questions particularly relevant, as the average was around 2.5. One of the reasons was given as an answer to a free text question: "*Sometimes it feels like it's too generic and doesn't apply to our work*". This could be the reason for why the team did not feel the relevance of the question. Another possibility is that the team felt that questions were unrelated if they rarely or never answered them. The question is, is the questions or types of code changes unrelated to the codebase if they are answered rarely, or are they unrelated because they have nothing to do with the product that the team is making. As mentioned in subsection 5.3.2, the security experts found the information valuable, even though the team always answered 0 to a few questions, because it gave confidence that it had not happened. All in all, it is unclear weather or not this requirement is fulfilled, but it is at least partially filled as at least some of the types of code changes are inspired from other publications.

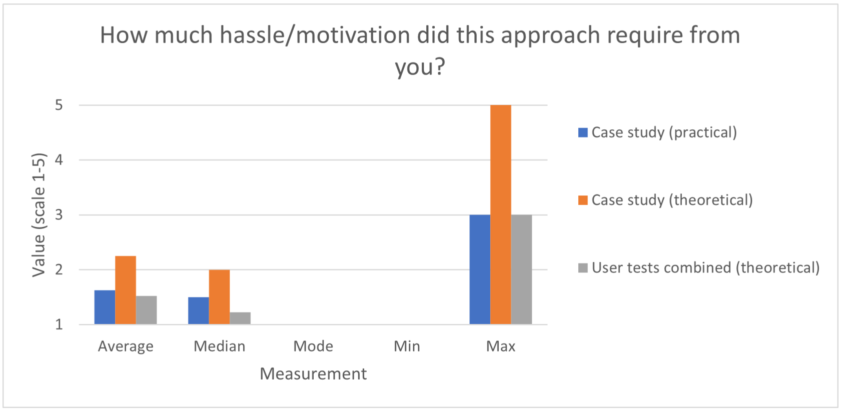


Figure 5.8: How much the extra hassle or motivation does the participants think this artifact requires, on a scale from 1 to 5. The data is from both the case study and the normalized data from the two paper prototypes. The data is about the theoretically described artifact and the implemented one used in the case study.

NFR3 is clearly fulfilled as the method-artifact exists within the version control systems, and the implementation shows that this is possible by creating it with GitHub actions and workflows, as described in section 4.3.

NFR4 can partially be answered through an argument. As seen in Table 4.2, the questions are mostly written without the use of security related terms. But, this can also partially be evaluated by asking the participants if they understood the questions. It is however important to note that this questions also is affected by the way the questions are formulated and their ambiguity. From Figure 5.10 it is clear that the team in the case study understood most questions, as 75% answered that they understood all the questions, which is in direct contrast to the user tests where 70% reported that they did not understand all the questions. It is important to note that this question captures if a participant understood all questions, and they have answered *no* even if they were unsure of a single question. It makes sense that the participants in the user tests had worse understandings of the questions. It could be because they were only introduced to them once, and were quickly, and it is logical that the understandings would increase as the questions are answered multiple times and discussed with teammates. This is one of the reasons for why the practical implementation of artifacts and techniques is so important, as commented in the pre-study [11]. All in all, the requirement is fulfilled, but there are room for improvements.

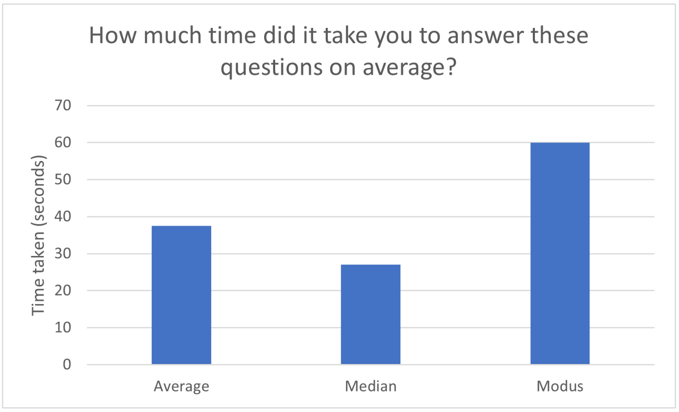


Figure 5.9: How long time the participants reported that it took to fill out the questions during pull requests, from the case study.

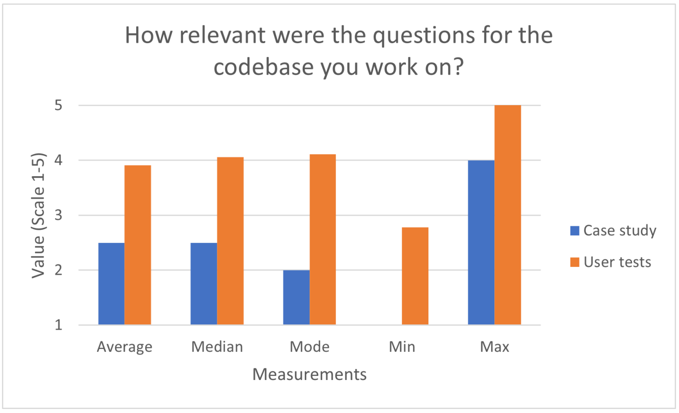


Figure 5.10: How relevant the participants found the questions from the artifact in relation to the code base they were working on, on a scale from 1 to 5. The data is from both the case study and the normalized data from the two paper prototypes.

5.3.5 Compare artifacts to other

To compare this artifact to other similar artifact, searches has been conducted on the internet. The most similar results is the publication from SAFECode [16], but this is only a list of recommendations, not a complete artifact like the one presented in this thesis. In addition, the recommendations from SAFECode[16] has been used as inpiration for parts of this artifact. It shares some similarites, but the recommendations from SAFECode[16] only contains the raw recommendations, and not the artifact that increases the usability and usefulness.
During the search I came across one tool [37] that could be similar to the artifact presented here, but it was not possible to gather any additional information about it.
If this method artifact is implemented using questions in the form of a checklist, like the one tested in the case study, then it has resemblance with other security activities that is dependant upon checklists, like

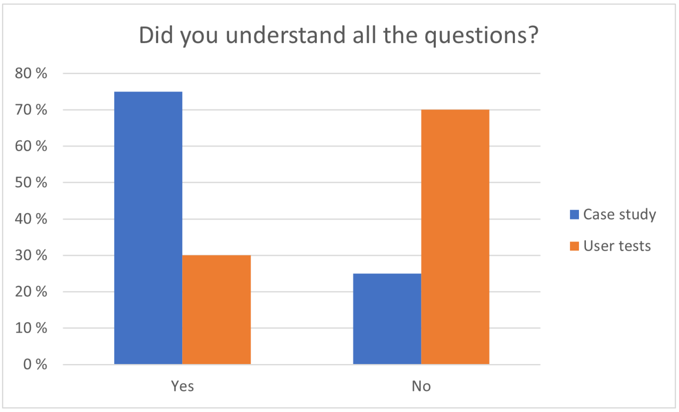


Figure 5.11: How many participants understood all the questions from the pull request forms. The data is from both the case study and the normalized data from the two paper prototypes.

the Autodesk continuous threat modeling methodology [6]. Both the Autodesk-approach and this artifact uses questions that ask the developers about *what* they have done, but the Autodesk-approach immediately recommends follow-up features that should be implemented, while this approach uses the data for analysis and then triggering manual threat modeling sessions [6].

During the user tests and the interviews, none of the participants said that they have tried anything like this. This comes as no surprise, as I have not found any other artifact that has the same goal, as presented above. This clearly indicates the need for further research in this area.

5.3.6 Investigate side-effects

None of the 10 participants in the user tests believed this would change the size of their PRs, and none of the eight participants in the case study experienced any change in the size of their PRs.
During both user tests and the interviews, the participants provided more open-ended answers, and the ones related to side-effects are presented below:

- A few participants were worried that some developers might not answer the questions correctly, and just "rush" through them
- The adoption of such an artifact would need to be handled in a structured and clear way, so that it is not pushed on a tech-leads.
- Sometimes it might be a few days between the beginning of a PR, and when it is ready for review, these questions might affect this somehow.
- Concerns were raised that this artifact quickly could become just another compliance activity.
- This artifact introduces more work, is another thing developers will have to do when they create PRs, and other developers will need to double-check the answers when they review the PR.
- One developer raised the concern that automation of this artifact could potentially collide with automatic bots that reside inside the source control system.
- After the case study, the participants told that if they didn't complete the form immediately when creating a PR, then it failed the pipeline, which sent mail to code owners and the developers, which

was annoying.

As seen from the list above, there were not many potential side effects. The first two potential side effects are both related to the technology acceptance, and makes it clear that the adoption of such an artifact would need to be handled in a good way, so that everyone see the value it brings. The third point is quite interesting, and should be studied further. The fourth and fifth point confirms what I feared in the beginning of this project namely that this should not take much time from the developers, this is captured in NFR1 and NFR3. It is important for both future work and future research to keep this in mind, but it is natural to think that this will be improved with automation. The second last point is very interesting and is something that has not been given any thought throughout this thesis. Care must be taken to ensure that the introduction of this new artifact does not disrupt existing automation and bots. The last point is the same as the one presented in the usability results, but it is also important here. If this form is not filled out, this artifact fails, which makes the entire pipeline fail, spamming code owners and developers with emails. This is clearly a side-effect that should be handled.

5.3.7 Formative evaluation

This subsection contains formative evaluation results, or in other words, how to improve the artifact. The suggestions are based on the user tests and case studies interviews, in addition to feedback from other security experts and observations.

Firstly, there has been a few recommendations and suggestions related to technical improvements to the implementation used in the case study. These are often easier to implement compared to the suggestions for the method-artifact itself, as they often only include a few lines of code. One of these is to stop the artifact from failing the pipeline workflow, which leads to spam-emails to the developers and code owners. In addition, if a developer has answered the questions, and then made a new commit, then the existing answers could be removed, as they most likely will need to be updated.

Throughout the work on the artifact, automation has come up in every meeting with security experts, in almost every user test and interview. It is clear that this artifact would be improved with automation. There are several parts that can be automated. Firstly, the calculation of the PR contribution indexes and accumulated contribution index could be done automatically, and it could automatically be compared to a threshold and send notifications when a trigger occurs. Secondly, the types of code changes could be automatically detected. This would reduce the interaction between developers and the artifact, which would reduce the artifacts impact on their workflows and its hassle. In addition, the list of different types of code changes could be expanded, as there is no need to limit its size. This would make it possible to add triggers for changes that rarely occurs, but have high security value. If the detection were fully automated, there would be no problem of unrelated questions for front end. Automation is therefore an important step of improving this artifact.

The artifact could also have different questions based on the different parts of the code. This would make it possible to have different questions for front end and back end development, as many back end questions are unrelated for the front end. This could be taken one step further and also add different questions based on which team member has made the commits, as different team members will have different responsibilities, this was suggested during the interview at the end of the case study with the head of security development and a senior infrastructure engineer. This could also be used to have multiple different accumulated contribution indexes, so that one can have one for front end, another for back end, and maybe even one for infrastructure. This would make the threat modeling sessions smaller and it would be easier to involve the relevant developers. In addition, this could make sure that testers doesn't need to answer the questions when they create tests, or they could have their own, specific questions.

The artifact could also be improved by indicating which part of the code that created the trigger, as this could focus the analysis and modeling on the parts that actually changed.

The artifact could also be improved by adding a question about the nature of the PR, if it is a refactoring or something new, or a combination. This information could be used to add another factor to the contribution index, which could make the triggers more precise. As new features and modifications of old features doesn't necessary bring the same potential risk.
One could also add another index factor to the different parts of the code, so that more critical parts can have a higher factor than non-critical parts. This could also improve the triggers.

Finally, the PR contribution index of a PR could be inserted into the PR as soon as it is calculated. This will make the benefits of the artifact more visible to the developers, and it could also be used to increase the understandings of the risks they bring through their daily work. This could improve the security culture and improve software security.

Chapter 6

Discussion

This section contains answers and discussions of the research questions, discussions about the implications of this thesis, discussions about the limitations, threats to validity, and finally some reflections.

6.1 Research questions

RQ1: How can threat modeling be triggered based on code changes?
The answer to this research question is the artifact itself, as described in section 4.1, and as we have seen in this thesis, it can be used to trigger threat modeling sessions, based on code changes.
Different code changes can be categorized based on how they impact the architecture, as described in section 4.1. Each of these types of code changes can be given a contribution factor, based on how much each type of code changes impacts the architecture or otherwise bring risks to the system. Whenever a change in the codebase is made, which is often done through a pull request, the number of changes within each types of code change can be counted. The contribution index for that PR can then be calculated by multiplying the number of changes within each type of code change with its contribution factor, and summarize this for each type of code changes. Then this can be stored in a database together with a reference to that pull request. The contribution indexes for all the PRs can be summarized to create the accumulated contribution index. If this index surpasses a threshold, a notification can be made to inform about the need for a new threat modeling session. This should ideally be automated, but instead of counting the number of changes within each category, the developers can answer questions to capture this information manually. The questions, contribution factors, and threshold can be changed and adapted for different teams.

RQ2: What is the effect of the suggested approach from RQ1?
This artifact is a method artifact, as described in subsection 3.1.1, and it is therefore still only at method-level. But to be able to perform any evaluation, I ended up creating a simple manual paper prototype of it, as well as a instantiation as a GitHub action which implements most of its features. These are by no means implementations of this method-artifact, but they share enough of its features and interfaces, to allow for some comparisons and evaluations. But it is still important to note that we have not evaluated the artifact, but we try to generalize the knowledge we gain about these prototypes to gain knowledge about the artifact.

In section 5.3 I present the evaluation results of this artifact. The results are grouped after the goals of evaluation for the design science research method framework, presented in subsection 3.1.1. I will now discuss and compare these results in more detail.

One evaluation is about **to what extent the artifact is effective for solving the problem**, presented in subsection 5.3.2. The main result is that developers and testers believe it is good, and rates it between 3 and 4 on a scale from 1-5, dependant on if you talk about the paper prototype or the case study, or a more theoretical artifact, as seen in Figure 5.1. This is not as high as one could hope for, but it is clearly above the average score of 3, which is good. We can also note that the scores were higher when we talked about the theoretical artifact. This is natural as the theoretical description contains more automation, and it would therefore require less work and hassle for the developers and testers. This number is a general rating of the entire artifact, and therefore includes its usefulness, effectiveness, efficiency, satisfaction, usefulness, and more. We need to view more detailed questions to gain a deeper understanding of what this number means and why it is as it is.
But before we move on to the details, it is important to note that this artifact is not designed for developers, but it is designed for security experts and managers, but in the manual version we are dependent upon the input from the developers. The developers are of course interested in creating high quality and secure software, but I believe that the responsibility of activities such as threat modeling is often placed upon security experts and managers. Therefore, the developers might not agree with security experts and managers when it comes to the usefulness of such an artifact.

The participants were also asked how useful they think this artifact is, both the one they tested and the theoretical one, and the results were almost identical to how they would rate this approach, this is presented in Figure 5.2. This is not surprising, as many would rate something based on how useful it is. The usefulness was ranged on average as between 3-4, by the participants, which was mainly developers and some testers. While the head of security development and senior infrastructure engineer ranked it as a solid 5. This is interesting, but is not surprising, as this is a tool for managers and security experts. But in addition to the accumulated contribution index and triggers, the head of security development and senior infrastructure engineer saw a massive potential for other use for the data generated from the artifact. This is very interesting and will be discussed further in future work and implications. Finally, it is interesting to see that most developers and testers thinks this artifact was worth the extra hassle and motivation, as seen in Figure 5.3. This could indicate that also the developers and testers feels the need to bridge the gap between software security and threat modeling.

The **usability** of the artifact was evaluated, and the results is presented in subsection 5.3.3. The System Usability Scores (SUS) from both the user tests and the case study were above average, which at least indicates that the usability is not terrible. It is important to not put too much emphasis on the SUS and that is why I only conclude that the results are above average, and therefore not terrible. Further, we can see from the other usability results that the artifact is quite good, but has room for improvements. Most of the feedback is related directly to either the paper prototype or the simple GitHub action, and thus the feedback does not "hit" the method-artifact particularly much. But, some of the feedback is applicable, such as the importance of involving the team in the creation and maintenance of the questions/their interaction with the artifact. The participants spent on average less than 40 seconds whenever they answered the questions. This is quite short, considered that they tested a manual implementation of the artifact. One can imagine that a more automated version would take even less time.

It is not much to say about the artifact and its **requirements**, other than that we can confirm that most are fulfilled through the informed argument presented in subsection 5.3.4. But we can see that the artifact has achieved the most important non-functional requirement, namely that it should affect the developers agile workflow as little as possible, which is why this is NFR1. I believe that this is the key to bridging the gap between security activities and agile development.

All artifacts has **side effects**, and these have been investigated and the results presented in subsection 5.3.6. In the beginning neither I nor the security experts I discussed the artifact with could think of many side effects, but during the case study we uncovered some. The only side effect we predicted were a potential change in the size of the pull requests, but the results clearly indicate that this was not the case. This is actually a very good thing, as one of the main goals was to not impact the developers workflow. The artifact did however have some other side effects, and the most problematic was the fact that the artifact failed the entire pipeline, spamming developers and code owners with emails. This was a very unfortunate side effect, and I believe that this has negatively impacted the results from the case study, at least it would make me more negative towards an artifact. But on the other hand, I believe that the developers and testers were able to forgive these types of annoyances and ignore it when reviewing the artifact.

One final side-effect that was uncovered was only a potential side-effect, as we were not able to verify it, namely that an implementation of this artifact could collide with automated bots that reside within the source control systems. If this method artifact is implemented manually, like the paper prototype and the case study, then I think this is a potential critical side effect. Because it would be very disruptively if this new artifact ruined other automation. But, I believe that if this artifact is implemented with a high degree of automation and with a little care, then this side effect could be handled.
All-in-all this artifact did not have many side effects, but it could be that more would be uncovered if the case study lasted longer than three weeks.

Part of the evaluation of this artifact is to **compare it to other similar artifacts**, but I was not able to find any other artifacts that tries to achieve the same goal or in other ways have a close resemblance. SAFE-Code[16] presented suggestions and recommendations, but this is just a list of potential triggers. I believe that this indicates the need for this artifact, and that there is a clear gap. One question one would have to ask is of course if this could indicate that the problem is not so important as I think. Because it could be that no artifact is find, because the industry has found other good solutions to this problem. I will however argue that this is not the case, as we clearly can see that the industry like this approach and finds it useful, as shown in section 5.3. In addition, I have had discussions with various security experts and they all liked this artifact. One final point could be that other similar artifacts are hidden deep inside highly complex and expensive software security software. This could be the case, but when I searched the web, I did not find any certain indications of this, other than one company which had threat modeling trigger as a sales point. But I was not able to find any more information about this

6.2 Implications for research

This results of this thesis and the information it has generated has created some implications for future research.

One implication of this research is the continued need for more research of software security within the agile context. This is not a new implication, but throughout the work on this thesis, I have many times struggled with finding good sources and information about threat modeling in agile. I would therefore recommend more research on threat modeling in agile, but not just on new techniques for finding threats, but on techniques that are well adapted to agile and that brings security without sacrificing velocity, as mentioned in my pre-study [11].

Another implication is that the research on this or similar artifacts should be continued. This artifact has been overwhelmingly well received by developers and testers, but especially by security managers and experts. This clearly indicates the potential this type of artifact has for the industry. This thesis has presented and evaluated a method-artifact, but future research could improve the method and in time, create an instantiating-artifact which could provide much higher fidelity and usefulness.

Future research could also attempt to improve the artifact with focus on the automation. In this thesis, I have spent most of my time on the general functionality of the artifact, and therefore little time has been left for automation and other improvements. If this artifact is to be implemented, it will need this automation, otherwise it can quickly become a compliance activity that violates the agile principles.

In addition, future work could focus their attention on how this or similar artifacts could be adopted by a team, and the practical details with the adoption and maintenance of the questions, and the follow up of triggers. This might seem trivial, but as we found in the pre-study last semester [11], very few threat modeling techniques contains these details, which could make it more challenging for the industry.

It would also be very interesting if future research included a study on the different types of code changes that could be used as triggers for artifacts like the one presented in this thesis. That type of research is sorely needed, and could rely on both the theory of software architecture, but will also need to study the industry over longer periods of time. This type of research could greatly improve this artifact and inspire other types of artifacts, as a more structured and well founded list of types of code changes will be a great improvement.

6.3 Implications to practice

This thesis shows that the problem of *when* to perform threat modeling could be performed more systematic than once a year, or semi-randomly. The industry should use this opportunity to review their own routines for threat modeling and adapt something more systematic, like this artifact or something similar.

The research in this thesis would not be possible without the contribution from the industry, from SpareBank 1 utvikling and Visma. It is important for the industry to protect their interests and security details against potential attacks, but the software security research will not be possible without their contribution.

And it is in their interest that the software security is improved. I will therefore urge the software development companies to cooperate with future research. They could for example use the artifact presented in this thesis.

6.4 Limitations and threats to validity

This thesis is subjects to several limitations and threats to validity of the results, and it should therefore be read and understood in the light of these.

The most obvious limitation is that this is a master thesis performed by a single student over the period of a single semester. This greatly limits the available time and resources for every activity. Especially when it comes to the extent of the data gathering. There were performed two initial user tests to get some initial data, and then a single case study over the period of three weeks, with a single development team. This means that the population for all the data generation is small, which is the reason that this thesis does not contain advanced statistical analysis. In addition, the case study should have been performed over longer periods of time to ensure that the data that I gathered were not a special case. Further, the case study should ideally have been performed in several iterations, so that the feedback and suggestions from one could be used to make improvements before performing the next iteration. But this was not possible within the limited time. The case study should also be repeated at multiple different teams and different companies, to ensure a better breath in the results.

As a student, I am not part of the company I performed the case study at, which means that I do not have any privileges or rights at the company, for security reasons. This is understandable, but it does make research more challenging, and it would not have been possible without exceptional help from security experts and managers at the company. But, this means that as a researcher, I have been distant from the case study when it happened, and it was not possible to observe the use of the artifact due to security and privacy. This means that observational data from the case study is missing, and also that the follow up of the case study was worse than it could have been. This might have negatively impacted the results, and lead to situations where I only learned about the failing of the pipelines and spamming emails after the case study was completed. The advantage, is that I do not have relations with any of the participants, which reduces potential positive result bias.
In addition, the team should have been better included in the creation and adaptions of the questions for their team before the case study started, but this was not possible because of time limitations from my own side and my limited access, and the time constraints of the company, as their participation were mostly after business hours.

Finally, this artifact is a method-artifact, and is thus theoretical. It is very challenging to evaluate a theoretical artifact, and the evaluation performed in this thesis is done by generalizing results from testing on paper prototypes, a simple GitHub action, and by discussing theoretical artifacts with the participants. This makes the evaluation weaker.

6.5 Reflections

There exists very few publications and sources with suggestions and recommendations for when to trigger threat modeling based on code changes, as discussed in the section 2.4. This made the work on this thesis challenging, as much of the work needed to be done from scratch without any good sources to rely on. The discussions with the security experts was especially useful in this regard, as they could provide experiences from the industry that could serve as replacements for the lacking academic sources needed this artifact. This means that this artifact is not so rooted in the academic studies as one would like, and perhaps a bit more based on the industry and their experiences. At the same time, the fundamental principles of this artifact is in line with at least parts of the software security theories, such as the results from Bernsmed et al. [15], Oueslati et al. [20], the pre-study last semester [11], and the Autodesk continuous threat modeling presented in [6].

Throughout this thesis I have had meetings with security experts from many different companies and countries, as described in subsection 3.3.3. In the beginning of the work on this thesis I had a rough idea about how the method-artifact could function, but I was very unsure if this had potential or if it was a poor idea. The discussions with my supervisor and security experts from both SpareBank 1 utvikling and Visma was crucial at this stage of the thesis. Through discussions, I gained confidence in the artifact, and it was further improved with their feedback and thoughts. Throughout parts the thesis I had weekly meetings with two senior infrastructure engineers that contributed to the work on the artifact, through brainstorming, feedback, and implementation suggestions. I do not believe that the implementation would be anywhere near its current state without their help. I believe that this type of academic-industry cooperation is essential for master thesis like this one, because this is a problem in the industry, and it should therefore be solved with the industry. I was later introduced to Matt Konda, and his feedback as an independent security expert in relation to this thesis was of great use. Both the confidence I gained from the meeting with him, as well as the multiple suggestions he presented improved the artifact, and several of the recommendations for future work on this artifact is inspired from him. Towards the end of the project I was also able to present this artifact to Rohini Sulatycki. She had nothing to do with this thesis, much like Matt Konda, and she also provided more confidence in this artifact, and some suggestions. Throughout the work on this thesis, it became clear to me that the security experts liked this artifact, which could indicate the need the industry has with triggering threat modeling. As we discussed in the section 2.4, little to no recommendations for triggers based on code changes exists, and this could be a reason for their confidence in this artifact. This shows the great importance for more research in this direction.

Chapter 7

Conclusion

7.1 Conclusion

This thesis aimed to improve software security in agile development by proposing a new method-artifact that can indicate when it is time to perform threat modeling based on code changes. More specifically, it answers the following research questions:
RQ1: How can threat modeling be triggered based on code changes?
RQ2: What is the effect of the suggested approach from RQ1?

The effect of the method-artifact was evaluated by generalizing data from two paper-prototyping user tests at SpareBank 1 Utvikling and a case study at Visma, which tested a minimum-viable-product implementation of the artifact. The minimum-viable-product implementation required the developers to manually categorize their code changes and input this into the artifact.

The developers and security experts involved in the tests agreed that this artifact effectively solved the problem. The developers found the artifact useful and rated it above average, while the security experts that oversaw the case study were very positive and found it highly useful and effective. This is natural, as this is a tool for security experts, but since the implementation was manual, the developers needed to provide it with data. In addition, the security experts saw great value in the raw output of the artifact as well, as it could be used to provide confidence in the work they had done, and it is also possible that it can be used to synthesize data about how a development team works.

The usability of the artifact is evaluated as good, with an average SUS score above average and within the acceptable ranges. On average, the participants of the case study spent 38 seconds on answering the questions whenever they created a pull request, which is short. This indicates that even the manual implementation did not impact their workflow substantially. Most of the usability challenges were related to technical details of the implementation and are therefore not particularly relevant to the method-artifact itself. Most participants agreed that the artifact would greatly benefit from automatic detection of code changes instead of manual input. Some side-effects were found, but these were minor annoyances in the implementation, not in the method-artifact itself.

To conclude, threat modeling in agile development can be triggered based on the code changes in a systematic way that is useful, usable, with few side effects, highly modifiable, and impacts agile development to a small degree. It is well received by both security experts and developers in the industry. It can be achieved by following the principles of the method-artifact or the instantiation proposed and evaluated in this thesis.

7.2 Future work

This thesis have identified and discussed multiple directions for future work. Future work could build upon both the method-artifact and the minimum-viable-product implementation presented in this thesis.

Automation has been mentioned as highly importance for this type of artifact, both by the researcher, other literature, and most of the participants. This artifact would greatly improve with more automation. Firstly, the code changes could be automatically detected and categorized. This would greatly reduce the interactions between the developers and the artifact, which would make it impact their agile development even less. In addition, the minimum-viable-product implementation could be extended to also include the automatic calculation of the contribution indexes and automatically create notifications on triggers.

The raw output of the artifact could be used to other purposes than triggering threat modeling. It could be possible to use them to create dashboards for the teams, where they can keep track of the various types of code changes, and perform analysis on how the team work.

There are few publications about potential triggers for threat modeling in agile, and future work could contribute in this direction. This could be used both to improve this method-artifact, but also ensure that the list of code changes is systematic and scientifically based.

This artifact was tested at two agile development companies, and the results are promising. However, future work could also repeat these evaluations and multiple other agile development companies to increase the validity of these results and identify more areas for improvement.

Bibliography

[1] O. Sviatun, O. Goncharuk, C. Roman, O. Kuzmenko, and I. Kozych, “Combating cybercrime: Economic and legal aspects,” *WSEAS TRANSACTIONS ON BUSINESS AND ECONOMICS*, vol. 18, pp. 751–762, Apr. 2021. DOI: 10.37394/23207.2021.18.72.

[2] Herjavec Group, "*2019 Official Annual Cybercrime Report*". 2020. [Online]. Available: https://www.herjavecgroup.com/the-2019-official-annual-cybercrime-report/ (visited on 05/29/2022).

[3] The Norwegian National Security Authority (NSM), *"Nasjonalt digitalt risikobilde 2021" (National Digital Risk 2021)*. 2021. [Online]. Available: https://nsm.no/aktuelt/nasjonalt-digitalt-risikobilde-2021.

[4] G. McGraw, “Software security,” *IEEE Security & Privacy*, vol. 2, no. 2, pp. 80–83, 2004. DOI: 10.1109/MSECP.2004.1281254.

[5] M. Dawson, D. Burrell, E. Rahim, and S. Brewster, “Integrating software assurance into the software development life cycle (sdlc),” *Journal of Information Systems Technology and Planning*, vol. 3, pp. 49–53, Jan. 2010. [Online]. Available: https://www.researchgate.net/publication/255965523_Integrating_Software_Assurance_into_the_Software_Development_Life_Cycle_SDLC.

[6] I. Tarandach and M. J. Coles, *Threat modeling: A practical guide for developing teams*. Sebastopol, California, USA: O’Reilly, 2020.

[7] A. Shostack, *Threat modeling: Designing for security*. Indianapolis, IN, USA: John Wiley amp; Sons, 2014.

[8] K. E. Wiegers and J. Beatty, *Software Requirements*, 3rd ed. USA: Microsoft Press, 2013.

[9] I. Sommerville, *Software engineering Global Edition*, 10th ed. Harlow, Great Britain: Pearson Education, 2015.

[10] K. Bernsmed, D. S. Cruzes, M. G. Jaatun, and M. Iovan, “Adopting threat modelling in agile software development projects,” *Journal of Systems and Software*, vol. 183, 2022. DOI: 10.1016/j.jss.2021.111090.

[11] K. H. Håkonsen and V. Ahmadi, “Threat analysis in agile,” Department of Computer Science, Norwegian University of Science and Technology, Tech. Rep., 2021 (Unpublished).

[12] K. Tuma, G. Calikli, and R. Scandariato, “Threat analysis of software systems: A systematic literature review,” *Journal of Systems and Software*, vol. 144, pp. 275–294, 2018. DOI: 10.1016/j.jss.2018.06.073.

[13] P. Torr, “Demystifying the threat modeling process,” *IEEE Security Privacy*, vol. 3, no. 5, pp. 66–70, 2005. DOI: 10.1109/MSP.2005.119.

[14] V. Drake, *Threat modeling*. [Online]. Available: https://owasp.org/www-community/Threat_Modeling (visited on 06/02/2022).

[15] K. Bernsmed and M. G. Jaatun, "Threat modelling and agile software development: Identified practice in four norwegian organisations," in *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2019, pp. 1–8. DOI: 10.1109/CyberSecPODS.2019.8885144.

[16] SafeCode, "Tactical threat modelling," *SafeCode*, 2017. [Online]. Available: https://safecode.org/wp-content/uploads/2017/05/SAFECode_TM_Whitepaper.pdf.

[17] S. Morgan, "Cybersecurity jobs report: 3.5 million openings in 2025," *Cybercrime Magazine*, Nov. 2021. [Online]. Available: https://cybersecurityventures.com/jobs/.

[18] P. Johannesson and E. Perjons, *An Introduction to Design Science*. Switzerland: Springer Publishing Company, Incorporated, 2014. DOI: 10.1007/978-3-319-10632-8.

[19] T. D. Oyetoyan, M. G. Jaatun, and D. S. Cruzes, "A lightweight measurement of software security skills, usage and training needs in agile teams," *International Journal of Secure Software Engineering (IJSSE)*, vol. 8, no. 1, pp. 1–27, 2017. DOI: 10.4018/IJSSE.2017010101.

[20] H. Oueslati, M. M. Rahman, and L. b. Othmane, "Literature review of the challenges of developing secure software using the agile approach," in *2015 10th International Conference on Availability, Reliability and Security*, 2015, pp. 540–547. DOI: 10.1109/ARES.2015.69.

[21] D. Soares Cruzes, M. Gilje Jaatun, K. Bernsmed, and I. A. Tøndel, "Challenges and experiences with applying microsoft threat modeling in agile development projects," in *2018 25th Australasian Software Engineering Conference (ASWEC)*, 2018, pp. 111–120. DOI: 10.1109/ASWEC.2018.00023.

[22] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd. IN, USA: Addison-Wesley Professional, 2012.

[23] D. Pandit, *Threat modeling: The why, how, when and which tools*, 2018. [Online]. Available: https://devops.com/threat-modeling-the-why-how-when-and-which-tools/ (visited on 06/02/2022).

[24] R. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Springer Berlin, 2014. DOI: 10.1007/978-3-662-43839-8.

[25] I. Burazin, "Rising stars of the tech world: Why developers are job market royalty," *Forbes*, Sep. 2021. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2021/09/09/rising-stars-of-the-tech-world-why-developers-are-job-market-royalty/ (visited on 06/05/2022).

[26] H. Sharp, Y. Rogers, and J. Preece, *Interaction design: Beyond human-computer interaction*. Hoboken, NJ, USA: John Wiley & Sons Inc, 2019.

[27] International Organization for Standardization (ISO), "Ergonomic requirements for office work with visual display terminals (vdts)," International Organization for Standardization, Geneva, CH, Standard ISO/IEC TR 9241-11:1998, Mar. 1998.

[28] J. Brooke, "Sus: A quick and dirty usability scale," *Usability Eval. Ind.*, vol. 189, Nov. 1995. [Online]. Available: https://www.researchgate.net/publication/228593520_SUS_A_quick_and_dirty_usability_scale.

[29] B. Oates, *Researching Information Systems and Computing*. London, Great Britain: SAGE Publications, 2005.

[30] R. Colin and K. McCartan, *Real World Research, 4th Edition*. Dec. 2017, ISBN: 978-1-118-74523-6.

[31] S. S. Stevens, "On the theory of scales of measurement," *Science*, vol. 103, no. 2684, pp. 677–680, 1946. DOI: `10.1126/science.103.2684.677`. eprint: `https://www.science.org/doi/pdf/10.1126/science.103.2684.677`.

[32] NTNU, *Collection of personal data for research projects - kunnskapsbasen - ntnu*. [Online]. Available: `https://i.ntnu.no/wiki/-/wiki/English/Collection+of+personal+data+for+research+projects` (visited on 06/09/2022).

[33] *Lov om behandling av personopplysninger (personopplysningsloven) (the personal data act)*, 2018. [Online]. Available: `https://lovdata.no/dokument/NL/lov/2018-06-15-38/KAPITTEL_gdpr#KAPITTEL_gdpr`.

[34] Norsk senter for forskningsdata (NSD). "Notification form for personal data." (), [Online]. Available: `https://www.nsd.no/en/data-protection-services/notification-form-for-personal-data`.

[35] H. Kniberg, *Scrum and XP from the Trenches: Enterprise Software Development*. USA: Lulu.com, 2007.

[36] MySQL, "Mysql 8.0 reference manual :: 13.2.6.2 insert ... on duplicate key update statement," *dev.mysql.com*, [Online]. Available: `https://dev.mysql.com/doc/refman/8.0/en/insert-on-duplicate.html` (visited on 06/11/2022).

[37] Apiiro, "Devsecops orchestration," *piiro.com*, [Online]. Available: `https://apiiro.com/ssdlc-processes-and-tools-orchestration/` (visited on 06/12/2022).

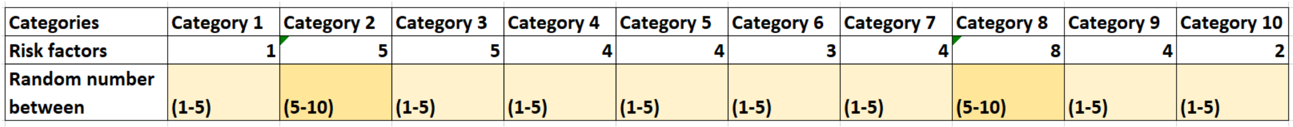
Appendix A

Artifact demonstration

This chapter contains a practical example of the theoretical case used to demonstrate the artifact in subsection 4.2.1. The demonstration is a very simple demonstration of the artifact. The demonstration shows that if you continue to add numbers to a value, it will sooner or later reach a threshold. This is the reason for the very short demonstration in section 4.2, and also why this more detailed description is placed here in the appendix. However, I try to show how the different elements relate to each other, such as the categories, contribution factors, PRs, PR contribution indexes, and accumulated contribution index, and this therefore brings some value for the interested reader.
This section begins with a complete table of the categories, the PR information, and then finishes with the calculations.

A.1 The demonstration categories

10 categories were chosen for the demonstration case, as it is about as many categories or questions as were used during the work on this thesis. In real life, each category or question will have a contribution factor chosen by the team. I believe that most categories will have low, similar values. This is because I suspect that the teams will not have the confidence or the knowledge to change them, or to determine the contribution factors of different categories or questions. At the same time, I believe that one or two will have substantially higher contribution factors than others, as most developers and security experts can agree that something is more risky than other.
An excel sheet was created with the 10 categories, and a random number were chosen for each. For eight of them, a number between 1 and 5 where chosen, and for two of them, a number between 5 and 10. The numbers were generated by using the "RANDBETWEEN" function in excel. So the formula for each category, except 2 and 8 is: "*=RANDBETWEEN(1;5)*", while 2 and 8 has "*=RANDBETWEEN(1;5)*". This is illustrated in Figure A.1.It is important to note that all the randomly generated numbers change whenever the excel document change.

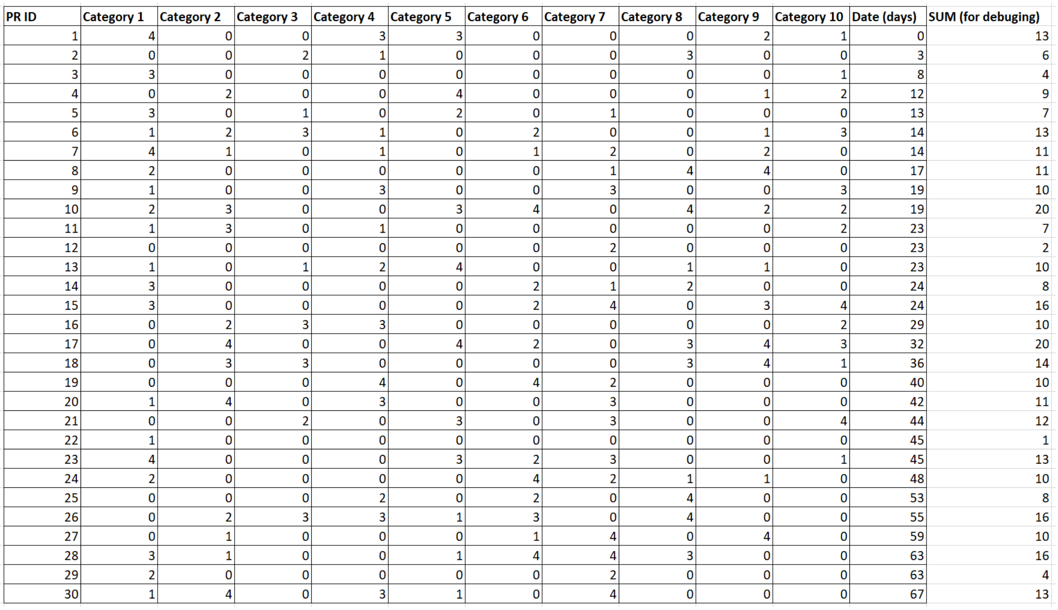


Categories	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8	Category 9	Category 10
Risk factors	1	5	5	4	4	3	4	8	4	2
Random number between	(1-5)	(5-10)	(1-5)	(1-5)	(1-5)	(1-5)	(1-5)	(5-10)	(1-5)	(1-5)

Figure A.1: The demonstration 10 categories, with randomly generated numbers. Category 2 and 8 have a number between 5 and 10, while the rest has between 1 and 5. Then numbers were generated with the "RANDBETWEEN" function in excel.

A.2 The demonstration pull requests

The pull request data represents 30 pull requests. It is challenging to determine the different values, as all development teams are different. The most important part was that each PR needed to have different values for the different questions, the values should not be too high, and it should be random.
To represent this, the value for each category in each PR is generated by first randomly generating 0 or 1, and then multiplying it with a random number between 0 and 5. This was done by first generating a number between 0 and 1, and then rounding it to a whole number, and then multiply it with a random number between 0 and 5. The formula for each category is: "*=ROUND (RAND();0)*RANDBETWEEN(0;4)*"
The time between the PRs are demonstrated by assigning the first PR to day 0, and then adding a random number of days between each pr, between 0 and 5. The formulate for calculating the day for a PR is therefore: "*=+the_date_of_the_previous_pr + RANDBETWEEN(0;5)*".
These values are illustrated in Figure A.2. It is important to note that all the randomly generated numbers change whenever the excel document change.



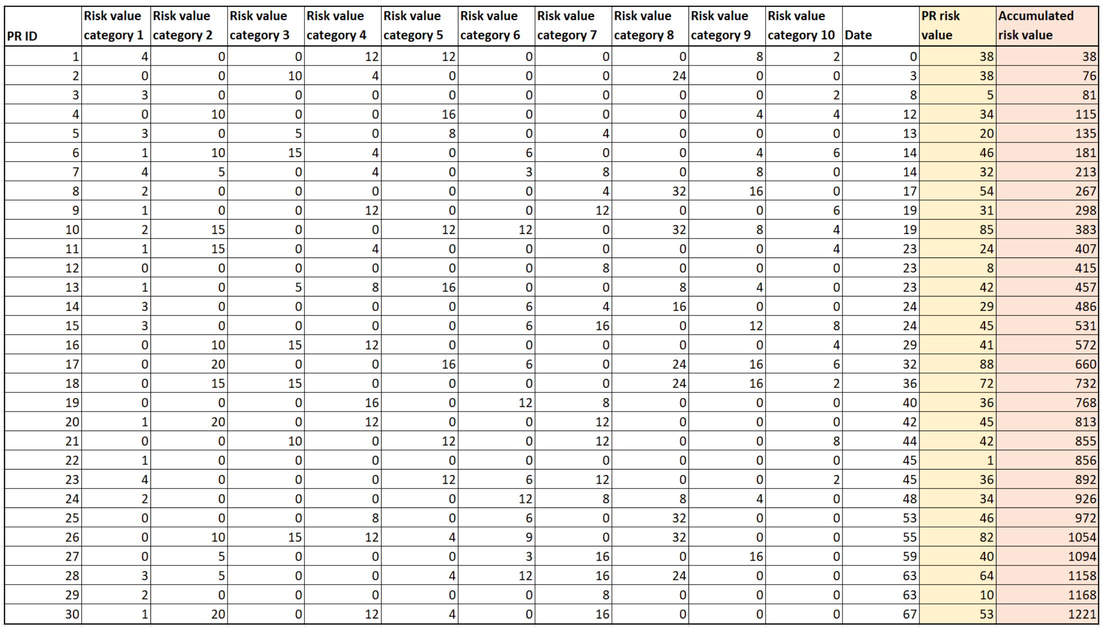
PR ID	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8	Category 9	Category 10	Date (days)	SUM (for debuging)
1	4	0	0	3	3	0	0	0	2	1	0	13
2	0	0	2	1	0	0	0	3	0	0	3	6
3	3	0	0	0	0	0	0	0	0	1	8	4
4	0	2	0	0	4	0	0	0	1	2	12	9
5	3	0	1	0	2	0	1	0	0	0	13	7
6	1	2	3	1	0	2	0	0	1	3	14	13
7	4	1	0	1	0	1	2	0	2	0	14	11
8	2	0	0	0	0	0	1	4	4	0	17	11
9	1	0	0	3	0	0	3	0	0	3	19	10
10	2	3	0	0	3	4	0	4	2	2	19	20
11	1	3	0	1	0	0	0	0	0	2	23	7
12	0	0	0	0	0	0	2	0	0	0	23	2
13	1	0	1	2	4	0	0	1	1	0	23	10
14	3	0	0	0	0	2	1	2	0	0	24	8
15	3	0	0	0	0	2	4	0	3	4	24	16
16	0	2	3	3	0	0	0	0	0	2	29	10
17	0	4	0	0	4	2	0	3	4	3	32	20
18	0	3	3	0	0	0	0	3	4	1	36	14
19	0	0	0	4	0	4	2	0	0	0	40	10
20	1	4	0	3	0	0	3	0	0	0	42	11
21	0	0	2	0	3	0	3	0	0	4	44	12
22	1	0	0	0	0	0	0	0	0	0	45	1
23	4	0	0	0	3	2	3	0	0	1	45	13
24	2	0	0	0	0	4	2	1	1	0	48	10
25	0	0	0	2	0	2	0	4	0	0	53	8
26	0	2	3	3	1	3	0	4	0	0	55	16
27	0	1	0	0	0	1	4	0	4	0	59	10
28	3	1	0	0	1	4	4	3	0	0	63	16
29	2	0	0	0	0	0	2	0	0	0	63	4
30	1	4	0	3	1	0	4	0	0	0	67	13

Figure A.2: The demonstration random values for 30 PRs. The contribution index for each category is calculated by using the "RAND", "RANDBETWEEN", and ROUND functions. While the Date is generated by using "RANDBETWEEN". The last column is there to demonstrate the sum of changes for each pr is 20 or lower.

A.3 Demonstration calculations

The contribution index for each category within each PR is calculated, and summarized for each PR, to calculate the PR contribution index. To calculate the index for a category within a PR, we simply multiply the contribution factor for that category with the generated data for that category for that PR.
The accumulated contribution index is calculated for each pr, by adding the previous accumulated contribution index with the PR contribution index. This is shown in Figure A.3. Note that the randomly generated numbers change whenever the excel document is changed.
The accumulated contribution index plotted against the pr dates, and a threshold of 1000 is shown in Figure A.4.

From Figure A.3 and Figure A.4 we can see that the contribution index within each category is successfully calculated for each category within each pr, and the PR contribution indexes are successfully calculated for pr. The accumulated contribution index is also evolving for each pr. We can see that the accumulated contribution index surpasses the chosen threshold of 1000 at around day 54.
This was a very simple demonstration of the artifact. It showed that if you continue to add numbers to a value, it will sooner or later reach a threshold, but it also shows relations between different concepts. This is the reason for the very short demonstration in section 4.2, and also why this more detailed description is placed here in the appendix.



PR ID	Risk value category 1	Risk value category 2	Risk value category 3	Risk value category 4	Risk value category 5	Risk value category 6	Risk value category 7	Risk value category 8	Risk value category 9	Risk value category 10	Date	PR risk value	Accumulated risk value
1	4	0	0	12	12	0	0	0	8	2	0	38	38
2	0	0	10	4	0	0	0	24	0	0	3	38	76
3	3	0	0	0	0	0	0	0	0	2	8	5	81
4	0	10	0	0	16	0	0	0	4	4	12	34	115
5	3	0	5	0	8	0	4	0	0	0	13	20	135
6	1	10	15	4	0	6	0	0	4	6	14	46	181
7	4	5	0	4	0	3	8	0	8	0	14	32	213
8	2	0	0	0	0	0	4	32	16	0	17	54	267
9	1	0	0	12	0	0	12	0	0	6	19	31	298
10	2	15	0	0	12	12	0	32	8	4	19	85	383
11	1	15	0	4	0	0	0	0	0	4	23	24	407
12	0	0	0	0	0	0	8	0	0	0	23	8	415
13	1	0	5	8	16	0	0	8	4	0	23	42	457
14	3	0	0	0	0	6	4	16	0	0	24	29	486
15	3	0	0	0	0	6	16	0	12	8	24	45	531
16	0	10	15	12	0	0	0	0	0	4	29	41	572
17	0	20	0	0	16	6	0	24	16	6	32	88	660
18	0	15	15	0	0	0	0	24	16	2	36	72	732
19	0	0	0	16	0	12	8	0	0	0	40	36	768
20	1	20	0	12	0	0	12	0	0	0	42	45	813
21	0	0	10	0	12	0	12	0	0	8	44	42	855
22	1	0	0	0	0	0	0	0	0	0	45	1	856
23	4	0	0	0	12	6	12	0	0	2	45	36	892
24	2	0	0	0	0	12	8	8	4	0	48	34	926
25	0	0	0	8	0	6	0	32	0	0	53	46	972
26	0	10	15	12	4	9	0	32	0	0	55	82	1054
27	0	5	0	0	0	3	16	0	16	0	59	40	1094
28	3	5	0	0	4	12	16	24	0	0	63	64	1158
29	2	0	0	0	0	0	8	0	0	0	63	10	1168
30	1	20	0	12	4	0	16	0	0	0	67	53	1221

Figure A.3: The demonstrated calculation for the contribution factors of each category for each category, and the calculated PR contribution index value (yellow) and accumulated contribution index (red).

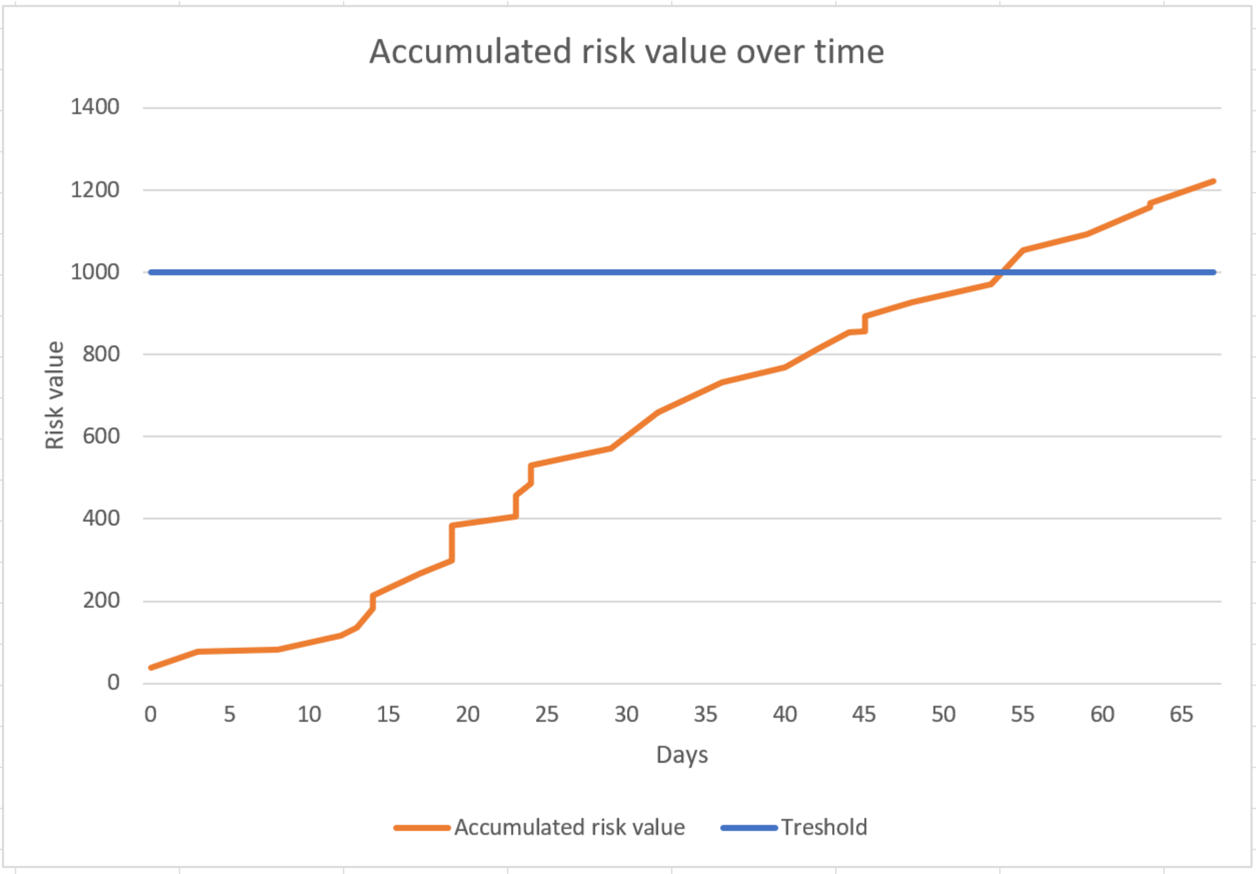


Figure A.4: The demonstration accumulated contribution index plotted against time, with a threshold of 1000.

Appendix B

NSD notification forms

This appendix contains the notification forms that the participants of both the user tests and case study used to gain information about their participation and give consent.

B.1 User tests

Are you interested in taking part in the research project *"Threat analysis in agile development"*?

This is an inquiry about participation in a research project where the main purpose is to improve software security in agile development. In this letter we will give you information about the purpose of the project and what your participation will involve.

Purpose of the project
Threat analysis and security review are areas with significant challenges in agile development, which negatively impact software security. This project will try to improve this integration/process by proposing a new method to gather some new data during code reviews to indicate the need for a new threat analysis session.

In this study, we will evaluate the usability for such an approach and will therefore gather some data from people working software development companies about their experiences with it.

This project is a master's thesis.

Who is responsible for the research project?
Norwegian University of science and Technology
Faculty of Information Technology and Electrical Engineering / Department of Computer Science

Why are you being asked to participate?
You are being asked to participate because you work as a developer, architect, security expert, software management, etc., in a software development company.

You are asked to participate, alongside other members of your company, as your company is interested in cooperating on this master's thesis. Your contact information has been obtained through your team leader.

What does participation involve for you?
As part of this study, you will test the new proposed approach for automatically triggering a threat analysis session. Your participation will be to participate in a user test of the new process. This should take no longer than 15 minutes.

All data will be stored on secure digital storage, agreed upon with your company.

Participation is voluntary
Participation in the project is voluntary. If you chose to participate, you can withdraw your consent at any time without giving a reason. All information about you will then be made anonymous. There will be no negative consequences for you if you chose not to participate or later decide to withdraw.

Your personal privacy – how we will store and use your personal data
We will only use your personal data for the purpose(s) specified in this information letter. We will process your personal data confidentially and in accordance with data protection legislation (the General Data Protection Regulation and Personal Data Act).

- The master student, Kristoffer Håkon Håkonsen, and the advisor, Daniela Soares Cruzes, will have access to the data.
- All data will be stored with security features in agreement with your company, such as encryption, two-factor authentication, edit-log, etc.
- All the data will be anonymized from the very start. The only identifiers will be the company you work at and the following title that describes you best: developer, security expert, manager. I will not store a link between your name and your answers.
- Your participation signature and contact information will be stored securely separate from your answers on the NTNU SharePoint, which is protected by encryption, record of changes, multi-factor authentication, restricted access, etc. No local copies will be stored.

What will happen to your personal data at the end of the research project?
The project is scheduled to end on the 1. November 2022. Some data will be published as part of the master thesis of Kristoffer Håkon Håkonsen, but this data will be anonymized and reviewed by your company. All other data will be deleted.

Your rights
So long as you can be identified in the collected data, you have the right to:
- access the personal data that is being processed about you
- request that your personal data is deleted
- request that incorrect personal data about you is corrected/rectified
- receive a copy of your personal data (data portability), and
- send a complaint to the Data Protection Officer or The Norwegian Data Protection Authority regarding the processing of your personal data

What gives us the right to process your personal data?
We will process your personal data based on your consent.

Based on an agreement with the Norwegian University of science and Technology, NSD – The Norwegian Centre for Research Data AS has assessed that the processing of personal data in this project is in accordance with data protection legislation.

Where can I find out more?
If you have questions about the project, or want to exercise your rights, contact:
- **The master student:** Kristoffer Håkon Håkonsen, kristohh@stud.ntnu.no, +47 48110613
- **The supervisor:** Daniela Soares Cruzes, daniela.s.cruzes@ntnu.no, +47 94249891
- **Data Protection Officer at NTNU:** Thomas Helgesen, thomas.helgesen@ntnu.no, +47 93079038
- NSD – The Norwegian Centre for Research Data AS, by email: (personverntjenester@nsd.no) or by telephone: +47 55 58 21 17.

Yours sincerely,

Kristoffer Håkon Håkonsen (researcher) and Daniela Soares Cruzes (Supervisor)

Consent form

I have received and understood information about the project “Threat analysis in agile development” and have been given the opportunity to ask questions. I give consent:

- ☐ to participate in this project by participating in a user test (participant observation and interview)

I give consent for my personal data to be processed until the end date of the project, approx. the 1. November 2022.

(Signed by participant, date)

B.2 Case study

Are you interested in taking part in the research project *”Threat analysis in agile development”*?

This is an inquiry about participation in a research project where the main purpose is to improve software security in agile development. In this letter we will give you information about the purpose of the project and what your participation will involve.

Purpose of the project
Threat analysis and security review are areas with significant challenges in agile development, which negatively impact software security. This project will try to improve this integration/process by proposing a new method to gather some new data during code reviews to indicate the need for a new threat analysis session.

In this study, we will evaluate the usability for such an approach and will therefore gather some data from people working software development companies about their experiences with it.

This project is a master's thesis.

Who is responsible for the research project?
Norwegian University of science and Technology
Faculty of Information Technology and Electrical Engineering / Department of Computer Science

Why are you being asked to participate?
You are being asked to participate because you work as a developer, architect, security expert, software management, etc., in a software development company.

You are asked to participate, alongside other members of your company, as your company is interested in cooperating on this master’s thesis. Your contact information has been obtained through your team leader.

What does participation involve for you?
As part of this study, you will test the new proposed approach for automatically triggering a threat analysis session. Your participation will be to participate in a user test of the new process. This should take no longer than 15 minutes.

All data will be stored on secure digital storage, agreed upon with your company.

Participation is voluntary
Participation in the project is voluntary. If you chose to participate, you can withdraw your consent at any time without giving a reason. All information about you will then be made anonymous. There will be no negative consequences for you if you chose not to participate or later decide to withdraw.

Your personal privacy – how we will store and use your personal data
We will only use your personal data for the purpose(s) specified in this information letter. We will process your personal data confidentially and in accordance with data protection legislation (the General Data Protection Regulation and Personal Data Act).

- The master student, Kristoffer Håkon Håkonsen, and the advisor, Daniela Soares Cruzes, will have access to the data.
- All data will be stored with security features in agreement with your company, such as encryption, two-factor authentication, edit-log, etc.
- All the data will be anonymized from the very start. The only identifiers will be the company you work at and the following title that describes you best: developer, security expert, manager. I will not store a link between your name and your answers.
- Your participation signature and contact information will be stored securely separate from your answers on the NTNU SharePoint, which is protected by encryption, record of changes, multi-factor authentication, restricted access, etc. No local copies will be stored.

What will happen to your personal data at the end of the research project?
The project is scheduled to end on the 1. November 2022. Some data will be published as part of the master thesis of Kristoffer Håkon Håkonsen, but this data will be anonymized and reviewed by your company. All other data will be deleted.

Your rights
So long as you can be identified in the collected data, you have the right to:
- access the personal data that is being processed about you
- request that your personal data is deleted
- request that incorrect personal data about you is corrected/rectified
- receive a copy of your personal data (data portability), and
- send a complaint to the Data Protection Officer or The Norwegian Data Protection Authority regarding the processing of your personal data

What gives us the right to process your personal data?
We will process your personal data based on your consent.

Based on an agreement with the Norwegian University of science and Technology, NSD – The Norwegian Centre for Research Data AS has assessed that the processing of personal data in this project is in accordance with data protection legislation.

Where can I find out more?
If you have questions about the project, or want to exercise your rights, contact:
- **The master student:** Kristoffer Håkon Håkonsen, kristohh@stud.ntnu.no, +47 48110613
- **The supervisor:** Daniela Soares Cruzes, daniela.s.cruzes@ntnu.no, +47 94249891
- **Data Protection Officer at NTNU:** Thomas Helgesen, thomas.helgesen@ntnu.no, +47 93079038
- NSD – The Norwegian Centre for Research Data AS, by email: (personverntjenester@nsd.no) or by telephone: +47 55 58 21 17.

Yours sincerely,

Kristoffer Håkon Håkonsen (researcher) and Daniela Soares Cruzes (Supervisor)

Consent form

I have received and understood information about the project "Threat analysis in agile development" and have been given the opportunity to ask questions. I give consent:

- ☐ to participate in this project by participating in a user test (participant observation and interview)

I give consent for my personal data to be processed until the end date of the project, approx. the 1. November 2022.

____

(Signed by participant, date)

Appendix C

Evaluation details

C.1 Paper prototype user tests

This section contains information about the paper prototype user tests.

C.1.1 The paper prototype

The paper prototype of the developers manual interaction with the artifact is shown in Figure C.1

C.1.2 Interview guide

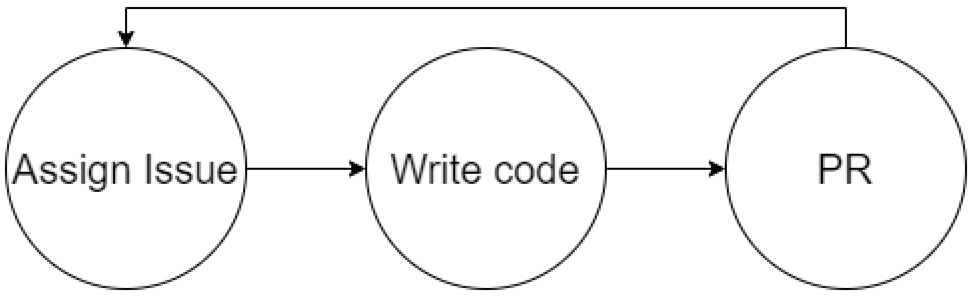
Below is the interview guide for the paper prototype user tests, the interviews were structured.

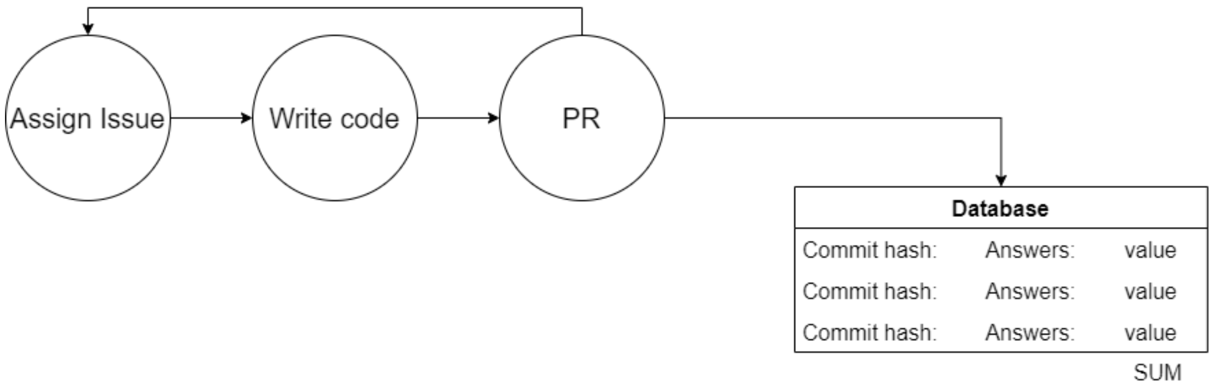
Interview guide

1. **Introduce myself**
2. **Describe the purpose of the test**
    a. The goal is to evaluate a new artifact, or process, as part of my master thesis.
    b. This test is a type of user test
    c. The goal of the new artifact is to improve the software security, without being at the expense of the agile values, without documenting for the sake of it, and without overburdening developers with work.
    d. The goal is to detect architectural drift, corrosion, and other code changes, that over time could create the need for threat modeling. To do this, we need input from the developers about what they have done in their pull requests. So that we can gather this data over longer periods of time and accumulate the changes and use them for analysis.
    e. Show images illustrating the new artifact
3. **Tell them that they can abort whenever they want**
4. **The goal of the test is not to evaluate you, but this new process. If any problems occur, they are because the artifact has faults, and it will not be because of you.**
5. **Ensure that they have read, understood, and signed the NSD forms.**
    a. Give opportunity to ask questions and explain if needed.
6. **Ask meta questions**
7. **Describe the equipment**
    a. For this test, we will use your last PR to answer some questions. Here is a form I have created. The form simulates the description of a PR and contains questions you need to answer.
8. **Teach how to think out-loud**
9. **Explain that I cannot help during the test**
10. **Ask if they have any questions before starting the test**
11. **Perform test**
    a. Read scenario
    b. Give printed-out form
12. **Ask “After the test questions”**
13. **Read theoretical description**
14. **Ask “After theoretical questions”**
15. **Fill SUS form**

16. Thank them for their time

Images illustrating the new artifact





Interview questions

Metadata:

1. What is your primary work title/task?
    a. Developer, tester, architect, designer, manager
2. How long have you worked as a developer/tester?
    a. less than 1 year, 1-5 years, 6-10 years, more than 10 years
3. Range your own security skills on range from 1 to 5. Where 1 is no skills and 5 is similar skils as as security expert.
    a. Scale 1-5
4. How important is security for you, when you develop? 1 = Features is most important. 5 = Security is the most important
    a. Scale 1-5

After the test

5. Do you have any immediate thoughts?
    a. Open-ended questions
6. What worked well?
    a. Open-ended questions
7. What worked bad?
    a. Open-ended questions
8. How relevant were the questions for the codebase you work with?
    a. Scale 1-5
9. Did you understand all the questions?
    a. Yes, no

Theoretical questions

10. How would you rate such an artifact, on a scale from 1-5?
    a. Scale 1-5
11. How much do you think such an artifact would impact your workflow?
    a. Scale 1-5
12. How much do you think such an artifact would slow down your development?
    a. Scale 1-5
13. How much hassle/motivation would such an artifact require from you?
    a. Scale 1-5
14. How useful do you think such an artifact would be?
    a. Scale 1-5
15. How much hassle/motivation would this new artifact require of you?
    a. Scale 1-5
16. Do you think it would change the size of your PRs?
    a. yes, no, unsure, neither

17. Did you like that the questions were grouped after new and modified/edited and then the qustions themselves are shorter (ALT A), or would you have them all written out fully (ALT B)? **(Only asked on the first user test)**
    a. Alt A , Alt B, unsure
18. Would you prefer if the questions were inside GitHub or BitBucket and with the type of format and answer type as this paper prototype test, or would you prefer to be sent to an external site with better formating and “smoother” answering mechanisms? **(Only asked on the first user test)**
    a. Inside GitHub/BitBucket, external site, unsure
19. Would you use such an artifact if it existed? **(Only asked on the second user test)**
    a. yes, no, unsure, neither
20. Would you be willing to test such an approach to evaluate and test its potential? **(Only asked on the second user test)**
    a. Yes, no, unsure, neither
21. Do you think this would be better or worse than similar security artifacts? **(Only asked on the second user test)**
    a. better, worse, unsure

Scenario

You have worked on an issue and you are done with all the code changes. You have pushed all the code changes to BitBucket/GitHub on its own branch, and you have just clicked the “Create pull request”-button. When you write the description of what you have done, you notice that a form with questions have been inserted.

Your task is: complete the form, using the data from your last pull request or the one you are currenly working on.

Theoretical description

Let's assume that you and your team have done this approach for over 3 months. You have refined the questions, removed the ones that are completely unrelated, added more, and tweaked them. In addition, you spend a few minutes (15) each other week to maintain the questions, if anyone has any feedback or suggestions. You can also assume that you have gone over the questions together, so there is no confusion about what they mean. In addition, everything that can be automatically detected is, which means that there are only around 2-3 questions to answer. You are still answering inside GitHub. The artifact creates triggers and notifies your team when it is time for threat modeling.

The user task

Questions:

Please answer the questions by inserting a numerical value after the ":" symbol or check the "☐ None of the above"-option.

I have created/added/opened/granted new:

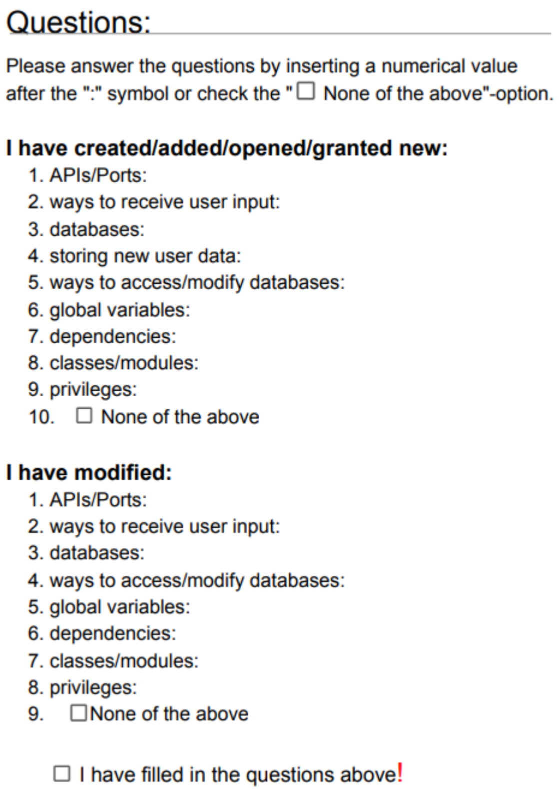
1. APIs/Ports:
2. ways to receive user input:
3. databases:
4. storing new user data:
5. ways to access/modify databases:
6. global variables:
7. dependencies:
8. classes/modules:
9. privileges:
10. ☐ None of the above

I have modified:

1. APIs/Ports:
2. ways to receive user input:

3. databases:
4. ways to access/modify databases:
5. global variables:
6. dependencies:
7. classes/modules:
8. privileges:
9. ☐ None of the above

☐ I have filled in the questions above!


Questions:

Please answer the questions by inserting a numerical value after the ":" symbol or check the "☐ None of the above"-option.

I have created/added/opened/granted new:

1. APIs/Ports:
2. ways to receive user input:
3. databases:
4. storing new user data:
5. ways to access/modify databases:
6. global variables:
7. dependencies:
8. classes/modules:
9. privileges:
10. ☐ None of the above

I have modified:

1. APIs/Ports:
2. ways to receive user input:
3. databases:
4. ways to access/modify databases:
5. global variables:
6. dependencies:
7. classes/modules:
8. privileges:
9. ☐None of the above

☐ I have filled in the questions above!

Figure C.1: Screenshot of the paper prototype used in the user tests

C.2 The theoretical description

Let's assume that you and your team have done this approach for over 3 months. You have refined the questions, removed the ones that are completely unrelated, added more, and tweaked them. In addition, you spend a few minutes (15) each other week to maintain the questions, if anyone has any feedback or suggestions. You can also assume that you have gone over the questions together, so there is no confusion about what they mean. In addition, everything that can be automatically detected is, which means that there are only around 2-3 questions to answer. You are still answering inside GitHub. The artifact creates triggers and notifies your team when it is time for threat modeling.

C.2.1 Results

The raw results of the two paper prototype user tests are presented in the following pdf, in Norwegian. Note that results from the first user tests, the first six rows, used a scale from 1 to 10, while the second uses a scale 1-5. The answers from the first scale is linear transformed to a scale on 1-5, as described in section C.7. It is also important to note that some questions were only part of user test 1 and some were only part of user test 2. User test 1 contained two questions about the form and layout, which is not in user test two, because the answers was used to improve the artifact before the second user tests, and it was therefore not necessary to have them there as well. User test 2 contains three additional questions, which is more about how the participants view the artifact and if they would be willing to use it.

Appendix: Combined raw data from two paper prototype user test (Page 1 of 7)

ID:	Q1	Q2	Q3	Q4	Annet
1	5-10 år	3,22	Utvikler	4,56	Frontend
2	Mer enn 10	2,78	Utvikler	2,33	Fullstack
3	Mer enn 10	2,78	Utvikler	3,67	Fullstack
4	Mer enn 10	3,22	Utvikler	3,67	Backend
5	Mer enn 10	4,56	Utvikler	3,22	backend
6	5-10 år	1,89	Utvikler	4,11	frontend
7	Mer enn 10 år	3	Utvikler	4	Backend.
8	Mer enn 10 år	3	Utvikler	3	Backend
9	5-10 år	3	Utvikler	3	Backend
10	1-4 år	4	Utvikler	4	Backend

Appendix: Combined raw data from two paper prototype user test (Page 2 of 7)

Notes	Time taken
Veldig liten PR, i går. Det må være avhukinger. Gjerne skript og sånt. Syntes det er veldig bra, men altfor langt, ikke gjort hvis det blir for langt.	30
Usikker på hva hen skal svare (hva er numerisk verdi?) Tenker at det å legge til en depedency er det samme som å modifisere. Flere av spørsmålene er jo statisk analyserbart, og burde kunne autooppdages.	40
Veldig usikker på spørsmålene, skjønner ikke dette med antall. Må tenke litt. Svarer nøyere enn de fleste. Usikker på hva som menes med noen spørsmål. Tenker litt.	100
Bruker lang tid på å tenke først. Tenker at dette er true/false, svart med 1 eller 0. SKjønner ikke at man kan svare noe annet enn 1 eller 0.	220
Veldig usikker på spørsmålene, føler at de er ja/nei (0/1). Litt usikker på privilegier. Svarer med veldig store tall: så den er ganske stor.	180
Vi hadde litt språkproblemer. Skjønte ikke helt spørreskjema. Svarte med haker (true/false), skjønte ikke at det var numerical med en gang, MEN skjønte det etterpå.	120
Usikker på om vedkommende skal svare med risikoen eller antall endrede	120
Usikker på hva dependencies er, ting man er avhengig av eller 3. parts bibliotek. - Syntes 8. har mye detaljer.	130
Litt usikker på noen spørsmål	130
Veldig kjedelig/liten PR	40

Q5
Folk kommer ikke til å skrive tekst. Ideelt sett burde mye blitt auto-lest av skripting. Created updated checkboxes. Bare checkboxes. Folk kommer ikke til å skrive. Usikker på om den skal ligge på PRen. Det kan bli lang tid immellom når man lager en PR og når man merger den.
Statisk analyse burde kunne finne masse av dette av seg selv. hen liker prinsippet. hen ser at dette kunne funket bra med github action. hen kunne gjort det fortere på en PC. Burde kunne gjort det fortere.
Dette likner på checklister, og de blir veldig sjeldent brukt. Når man må svare på sånne ting, om kvaliteten på svarene blir gode eller ikke? Og hvis svarene ikke er gode, hva kan man da bruke de til? hen er egentlig imot å innføre regler og andre tunge prosesser, fordi de ofte blir ikke blir brukt, de bare ligger der og er compliance, men at man ikke er bevist på verdien man får igjen fra de. hen tror også at de andre tiltakene man allerede har og at dette er en stor compliance aktivitet som ikke kommer til å funke spesielt godt for utviklere. hen føler at når man er svært erfaren utvikler og man er ganske god, så føles dette litt unødvending, men kanskje mer relevant for nyere utviklere. Kanskje mer relevant for nye. Kanskje kan brukes for å minne folk på at det finnes visse sikkerhetseleementer. Så dette er veldig avhengig av av teamet og risikoen som man opplever og har. Ville heller sendt utviklere på kurs og gjøre de bedre i stedet for slike aktiviteter. Viktigere å bygge flinke utviklere i stedet for prosess. Men at
hen tenker at en del av dette de har i sin arbeidshverdag, men kanskje litt mer systematisert. Og at dette er noe man burde ha i bakhodet, og dette gir strukturen. Dette er mer målbart. Kul ide, god nice.
Også registrerer hen at det blir ganske mye likt på første og andre delen (altså created/modified). Avhengigheter er veldig forskjellige, så det blir veldig lite nyansert å ha en felles score for alle dependencies. Så noen er lav risk og andre har høy risiko.
hen har aldri tenkt på sånt. Har aldri tenkt på sånne ting. Jobber mest frontend, så ikke så relevant. Også er jo bygget på en sikker arkitektur.
Vedkommende tenker at det har stort potensialet for automatisering. Det blir interessant å se hva resultatet blir. Det blir spennende å se om man klarer å estimere riktig, gleder seg til å se resultatene. Hen syntes det er veldig nyttig. Vi vet veldig sjeldent når. Vi sier at man skal oppdatere det "regelmessig", eller etter store endringer. -
Litt usikker på noen av spørsmålene - Stusset på antall nye klasser spørsmålet, burde blitt funnet ut automatisk - Eller syntes det var veldig bra og enkelt - Dette ville gått mye lettere når man har gjort det flere ganger, veldig nytt første - Ville kanskje lurt på hvorfor hen skulle gjort
Usikker på commit hashen, så det kan bli risikabelt. -> pga. rebasing. Kanskje gjøre det etter man har gjort merge Viktig at utviklerne får feedback, så det må ha en veldig god måte å få vite verdien på. -
Det kan være mulig å klassifisere oppdateringer og nye implementasjoner på forskjellig måte -> Burde være forskjellig risiko på å endre og legge til ny

Q6	Q7
Det er bra er at man får selvrefleksjon.	For langt, folk kommer ikke til å fylle. Må være enklere. Folk kommer bare til å huke av for none.
Likte selve prinsippet.	Over tid, hvis man innfører noe sånt, så ville man over tid irritert seg over statisk hentet ut. Det er irriterende å legge til noe man burde kunne scanne.
Les over	Les over
Hvis man får mye ut av verktæøyey uten å gi for mye så er det bra! Sikkerheten er jo en prioritet, man vil jo bruke minst mulig tid på administatrasjon og mest på features.	hen skjønner ikke helt hva hen skulle svare.
Tror at selve prinsippet kan gi en verdi, samtidig så kan det bli irritasjon. Eller at man kan svare gnaske raskt og uten å tenke.	Mye burde kunne automatiseres.
Nei	Nei
Fremtidig automatisering	Usikker på hvor vanskelig det er å autoamtisere
Tenker det var litt nyttig, å gi den veldig grove angivelsen av at hen la til et nytt API, men burde også kommet frem av beskrivelsen.	Trenger mer automatisering
Likte pinsippet	Bekymret for commit hash og at utviklere ikke fyller inn ordentlig
Veldig konkret. Har aldri tenkt over dette før	Likte ikke at endret og ny var sammen, føler at det medfører forskjellig risiko

Appendix: Combined raw data from two paper prototype user test (Page 5 of 7)

Q8	Q9	Annet3	Q10	Q11	Q12	Q13
4,11	Nei	Ble ikke en 10 på spørsmål 13, fordi ikke-tilpasset frontend. Må være mer spesifikt enn team.	3,67	1,89	1,44	1,44
2,78	Ja	Liker spørsmålene sånn generelt, mye bedre enn sånne overordenee compliance greier. Ikke så relevant fordi hen jobber med frontend akkuratt nå.	5,00	1,00	1,00	1,00
4,11	Nei	Opplever at spørsmålene kunne vært relevant for teamet, men ikke for hen spesifikt. De kan være relevante, men ofte endrer man ikke på det de spørsmålene spør om. 14. Var litt usikker.	2,78	1,44	1,44	1,89
4,11	Ja		4,11	1,00	1,00	1,00
3,22	Nei	Forstod ikke helt hva man spurte etter. Se notater under testen.	3,67	1,89	1,44	1,89
2,78	Nei	Tror mye skyldes språkproblemer	3,67	1,00	1,00	1,00
5	Nei	Kunne droppet å skrive 0-999, virker som man skal sette inn en skala ellerno	4	2	2	1
4	Nei	21. Det er veldig forskjell fra team til team og app til app	5	2	2	1
4	Nei		5	2	3	2
5	Ja		4	3	3	3

Appendix: Combined raw data from two paper prototype user test (Page 6 of 7)

Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21
4,56	Vet ikke	Uendret	ALT A	GitHub/ BitBucket	NA	NA	NA
4,11	Ja	Uendret	ALT B	Foretrekker 3. part	NA	NA	NA
2,78	Vet ikke	Uendret	Unsure	Foretrekker dette	NA	NA	NA
4,11	Ja	Uendret	ALT A	GitHub/ BitBucket	MA	NA	NA
3,67	Ja	Uendret	ALT A	GitHub/ BitBucket		NA	NA
3,67	Vet ikke	Uendret	Unsure	GitHub/ BitBucket	NA	NA	NA
3	Ja	Uendret	NA	NA	Ja	Ja	Vet ikke
5	Ja	uendret	NA	NA	Ja	Ja	Bedre
5	Ja	Uendret	NA	NA	Ja	Ja	Vet ikke
5	Ja	Uendret	NA	NA	Ja	Ja	Vet ikke

Annet4
16. Utfordringen er tidsaspektet, og man må skille på front og backend .20. Veldig teorietisk aufori. Dette må settes inn i ett system og det må ikke bare bli skubbet på techlead.For frontend, føles ikke dette så relevant ut. 23. Lage en listeformat der en kollonne og man kan huke hele veien 24. Må være integrert, må ikke få godkjent før fylt ut.
De har jo allerede en slags idee om en kodeendring burde kreve mer analyse, eller ikke. 17. Nesten alle tenker på vedlikehodlet av spørsmålene. 22. Kunne vært kult om folk ikke lagde for store PRer, sånn at man får noen advarsler hvis PRen blir for lang. Friksjonsvariabel, for hvor stor PRen er, og kanskje basert på statisk analyse. hen vil ha bot, som lager en post pr. spørsmål. hen ville foretrukket en webhook som gir statusen på skjemaet i github, men som tar deg til en 3. part for å fylle ut. Hvis det er smooth.
Veldig usikker på hele greia, sliter med å se det for seg. hen føler at man ikke burde trenge å svare 0, så vekk med none of the above.
Mange PR er veldig små og har ikke noe med sikkerhet å gjøre. Føles waste for frontend folk. Kanskje spørsmålene kan komme opp hvis man har endret på filer der sånt er viktig?
Aldri prøvd noe liknende
Tror mange er redde for slike spørsmål, siden det er sikkerhet. 28. Veldig avhengig av applikasjonen.
Bekymret for at dette vil kollidere med automatiske botter som automatisk lager og godkjenner Prer. I tillegg har litt vanskelig for å se det for seg, så svarer "alltid" 3.

C.3 Case study

C.3.1 Interview guide

The interview with the participants of the case study was a structured interview, is presented below.

Interview guide

This was a structured interview.

1. **Introduce myself**
2. **Describe the purpose of the test**
    a. The goal is to evaluate a new artifact you have tested for three weeks.
    b. Give quick recap of the artifact they have used
3. **Tell them that they can abort whenever they want**
4. **The goal of the test is not to evaluate you, but this new process. If any problems occur, they are because the artifact has faults, and it will not be because of you.**
5. **Ensure that they have read, understood, and signed the NSD forms.**
    a. Give opportunity to ask questions and explain if needed.
6. **Fill in SUS forms**
7. **Ask meta questions**
8. **Ask about the implementation they have tried**
9. **Read theoretical description**
10. **Ask theoretical questions**
11. **Thank them for their time**

Interview questions

Metadata:

1. What is your primary work title/task?
    Developer, tester, architect, designer, manager
2. How long have you worked as a developer/tester?
    less than 1 year, 1-5 years, 6-10 years, more than 10 years
3. Range your own security skills on range from 1 to 5. Where 1 is no skills and 5 is similar skils as as security expert.
    Scale 1-5
4. How important is security for you, when you develop? 1 = Features is most important. 5 = Security is the most important
    Scale 1-5

About the artifact they have tried

5. About how much time did it take you to answer these questions?
    Scale 1-300 seconds
6. What do you think about the artifact you have tested?
    Open-ended questions
7. What was good about the artifact you tested?
    Open-ended questions
8. What was bad about the artifact you tested?
    Open-ended questions
9. Did you understand all the questions on GitHub?
    Yes, no
10. How relevant were the questions for the codebase you work on?
    Scale 1-5
11. How would you rate this artifact?
    Scale 1-5
12. How much did this artifact impact your workflow?
    Scale 1-5
13. How much did this artifact slow down your development?
    Scale 1-5
14. How much hassle/motivation did this artifact require from you?
    Scale 1-5
15. How useful do you think this artifact was?
    Scale 1-5
16. Do you think this would be worth the hassle/motivation?
    Scale 1-5
17. Did this change the size of your pull requests?
    Scale 1-5
18. Do you think that this artifact was better or worse than similar activities you have tried?
    Scale 1-5
19. Is there anything else you would like to say about the artifact you have tested?

Open-ended

About the theoretical artifact

20. How would you rate such an artifact?
    Scale 1-5
21. How much do you think such an artifact would impact your workflow?
    Scale 1-5
22. How much do you think such an artifact would slow down your development?
    Scale 1-5
23. How much hassle/motivation would such an artifact require from you?
    Scale 1-5
24. How useful do you think such an artifact would be?
    Scale 1-5
25. Do you think this would be worth the hassle/motivation?
    Scale 1-5
26. Do you think this would change the size of your pull requests?
    Scale 1-5
27. Would you use such an approach if it was implemented?
    a. Yes, no, unsure
28. Would you be willing to test such an approach to see if it is worth it?
    a. Yes, no, unsure
29. Do you think that such an approach would be better or worse than similar activities you have tried?
    Better, worse, unsure, I have never tried anything like this
30. Do you have anything more to say about the theoretical approach?
    Open-ended

Theoretical description

The next questions are similar to the ones before, but now they are about the theoretical approach. Let's assume that you and your team have done this approach for over 3 months. You have refined the questions, removed the ones that are completely unrelated, added more, and tweaked them. In addition, you spend a few minutes (15) each other week to maintain the questions, if anyone has any feedback or suggestions. You can also assume that you have gone over the questions together, so there is no confusion about what they mean. In addition, everything that can be automatically detected is, which means that there are only around 2-3 questions to answer. You are still answering inside GitHub. This system calculates the indexes and creates triggers.

The second interview was an unstructured interview, where the outputs of the artifact was discussed with security experts from Visma. The outputs are shown in Appendix D. The interview guide is presented below.

Interview guide

This was an unstructured interview.

1. **Introduce myself**
2. **Describe the purpose of the interview**
   a. The goal is to evaluate the outputs of the artifact
3. **Tell them that they can abort whenever they want**

4. **Talk about the themes and show the corresponding tables and graphs:**

   - **The raw output of the artifact**
   - **The accumulated output, without contribution factors**
   - **The accumulated output with contribution factors**

C.4 The theoretical description

The theoretical description was similar to the one presented for the user tests, section C.2.

C.5 Results

C.5.1 Interview with participants

The raw results of the interview with the participants from the case study at Visma is presented in the following pdf, in Norwegian.

Appendix: Raw data from the interview with the participants of the case study (Page 1 of 4)

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7
1	Architect	More than 10 years	4	5	10	Small bug; does not see the checker when there is a whitespace in front of the "x".	It was easy to use it.
2	Developer	5-10 years	3	4	60	Easy to use. Doesn't bother every time I do PR.	Easy to use. Doesn't bother every time I do PR.
3	Architect	5-10 years	3	3	100	I think it is wonderful integrated but it will annoy the developers.	
4	Developer	5-10 years	2	3	34	Interesting, would be nice to see a risk score in the pr. can answers be provided automatically for some questions? Like how many new files were added	I see potential in it for future, like stop a pr if risk score is low
5	Developer	1-4 years	3	3	20	It's not used that often because those changes aren't that common. If you click it by accident few times it fails few times. Answers need to be easier to see	Keeps in check what has been done
6	Tester	1-4 years	1	1	1	for non developers is it a tiny minor inconvenience, as we will never answer the questions	helps devs see how much of their work could be a security risk, which raises awareness, increasing security concerns
7	Developer	5-10 years	3	3	15	It´s userful and easy to use.	Will help with the security
8	Developer	5-10 years	5	3	60	It's okay	

Appendix: Raw data from the interview with the participants of the case study (Page 2 of 4)

Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18
none	Yes	4	5	2	1	1	4	Yes	About the same	I have never tried anything like this
Most of questions are not really applicable to Frontend codebase.	No	2	4	1	1	2	4	Yes	About the same	I have never tried anything like this
	Yes	3	3	3	2	3	5	Yes	About the same	I have never tried anything like this
Reset answers after a new commit(sometimes I modified the answer but I don't know if it was saved/updated in the db)	Yes	3	3	1	1	2	4	Yes	About the same	I have never tried anything like this
Usually need to keep empty	Yes	2	2	1	1	1	1	Neither	About the same	I have never tried anything like this
when the pr is created it always fails on this artifact, "spamming" the dev's inbox daily	Yes	1	4	1	1	1	4	Yes	About the same	I have never tried anything like this
Is one more thing that you have to consider everytime	Yes	2	3	2	1	1	3	Yes	About the same	I have never tried anything like this
	No	3	3	2	1	2	2	Neither	About the same	I have never tried anything like this

Appendix: Raw data from the interview with the participants of the case study (Page 3 of 4)

Q19	Q20	Q21	Q22	Q23	Q24	Q25	Q26	Q27
no	4	1	1	1	4	Yes	About the same	Yes
	4	2	1	2	5	Neither	About the same	Unsure
no :)	5	3	4	5	5	Yes	About the same	Yes
	4	4	2	4	5	Yes	About the same	Yes
Sometimes feels like it's too generic and doesn't apply to our work usually	3	2	2	2	2	Yes	About the same	Unsure
it consistently fails after a pr was created, is there a way to delay the execution of this action? because right after i create the pr, I press the checkbox and I still get the email that it failed	5	1	1	1	5	Yes	About the same	Yes
	3	2	1	2	3	Neither	About the same	Unsure
Automatize it	4	1	1	1	3	Neither	About the same	Unsure

Appendix: Raw data from the interview with the participants of the case study (Page 4 of 4)

Q28	Q29	Q30
Yes	I have never tried anything like this	no
Yes	Unsure	
Yes	I have never tried anything like this	
Yes	I have never tried anything like this	
Yes	I have never tried anything like this	
Yes	Better	
Unsure	I have never tried anything like this	
Yes	I have never tried anything like this	

C.5.2 Interview with the head of security development and Senior Infrastructure Engineer

At the end of the case study, I performed an unstructured interview with Monica Iovan, Head of security development, and the senior infrastructure engineer Nicoleta Botosan, on the 25. of March 2022. The results are referenced in the report, but below are some quotes from the different themes:

The raw output of the artifact:
I find it very useful.
Can this work if we have different questions that contribute to the same sum, from different areas? For example, if we have 3 different lists of questions. One for front end, one for back end, and one for infrastructure. Would this formula work with the points from these three different sources? Both agree that this would be very good. Or, they could be stored in different tables, and created different triggers. This could make it much easier to call the correct people in for a threat analysis, based on which table triggered. So that we don't need to call in front end people if infrastructure triggered something. It could auto detect what code changes are done, or it could add all questions, and the teams can only answer the ones related to them.

The accumulated output, without contribution factors:
I think it is useful for multiple reasons: The first is the accumulated risk graph that will trigger the threat analysis. But also I think that based on the questions, we can find some patterns on how the team works, and improve the way the team works based on the patterns.

The accumulated output with contribution factors:
I think so, what could be a bit different for some of the questions, to add a question about if the code change was about refactoring, addition, or just adding something new. Because splitting a large class into smaller classes could be better. So, create a way to differentiate between if this is modification or something new, or both. So only one new question. The same goes for data classes.
We can also use this to estimate the velocity over time. As we can see how much the team did in one quarter, and then we can see next quarter if they have a high number of changes here. And then do some analysis. And measure a bit what they are working on, just by this data. So not only looking at tickets and story points, but at the actual security concerns and the complexity that has been introduced over time.

If you were to rate this artifact on a scale from 1-5, what would you give it?
10. I love this idea; I think it is really great! I think it is really amazing.
5. I think this idea is super awesome, and I was very thrilled to work on it. It is multiplied stuff we can continue to implement and continue to work on.

C.6 SUS form

The SUS form used during the evaluations, is shown in Figure C.2, and is from the publication by John Brook [28]. During the evaluations with Norwegian participants, it was translated to Norwegian.

C.7 Linear transformation scales

To transform the scale of 1-10 from the first paper prototype user test to a scale of 1-5, like in the second user test and the case study, linear transformation was used. It is done in two steps. Steps 1, convert the old scale into the 0-1 scale. Step 2, transform to the wanted scale.

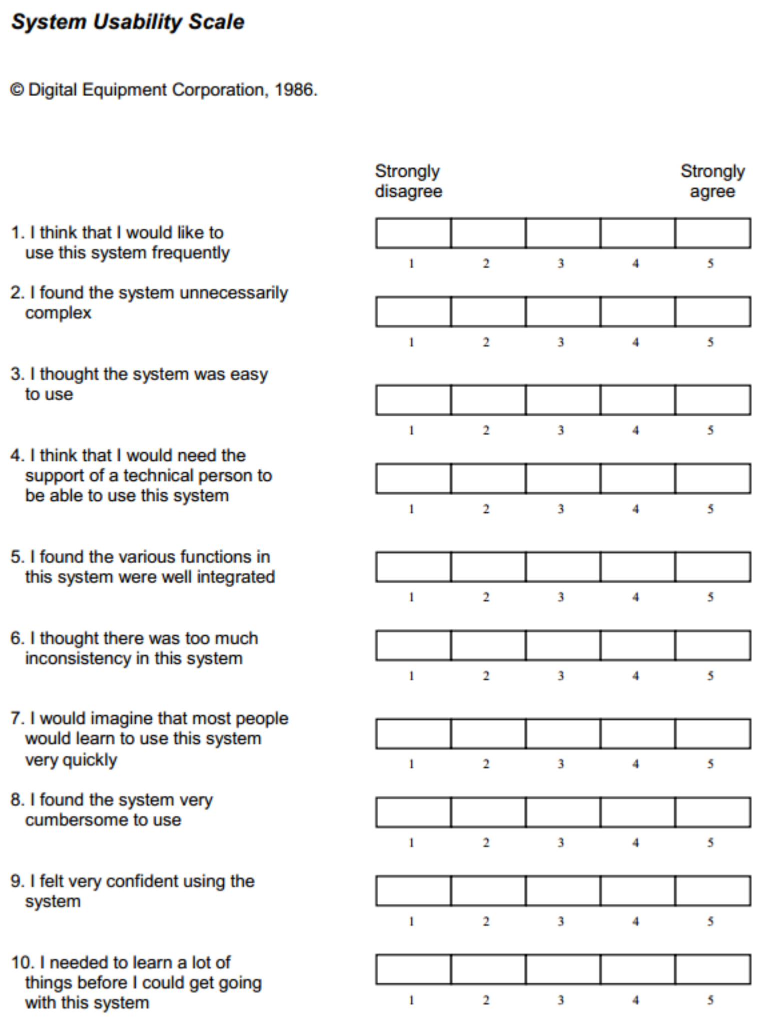
Step 1: Convert to a 0-1 scale
$tempNewNalue = (oldValue - minValueOldScale)/(maxValueOldScale - minValueOldScale)$

Step 2: Convert to a 1-5 scale
$newValue = tempNewValue * (maxValueNewScale - minValueNewScale) + minValueNewScale$

The formula for converting from 1-10 scale to 1-5 scale
To calculate from a 1-10 scale to a 1-5 scale, the combined formula becomes:
$newValue = ((oldValue - minValueOldScale)/(maxValueOldScale - minValueOldScale)) * (maxValueNewScale - minValueNewScale) + minValueNewScale$

We can insert numbers
$newValue = ((oldValue - 1)/(10 - 1)) * (5 - 1) + 1$
Which can be rewritten as:
$newValue = ((oldValue - 1)/9) * 4) + 1$

This formula was applied to all the results from the first paper prototype user test to convert the results.



System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

Figure C.2: SUS form used for the evaluation [From John Brooke [28]]

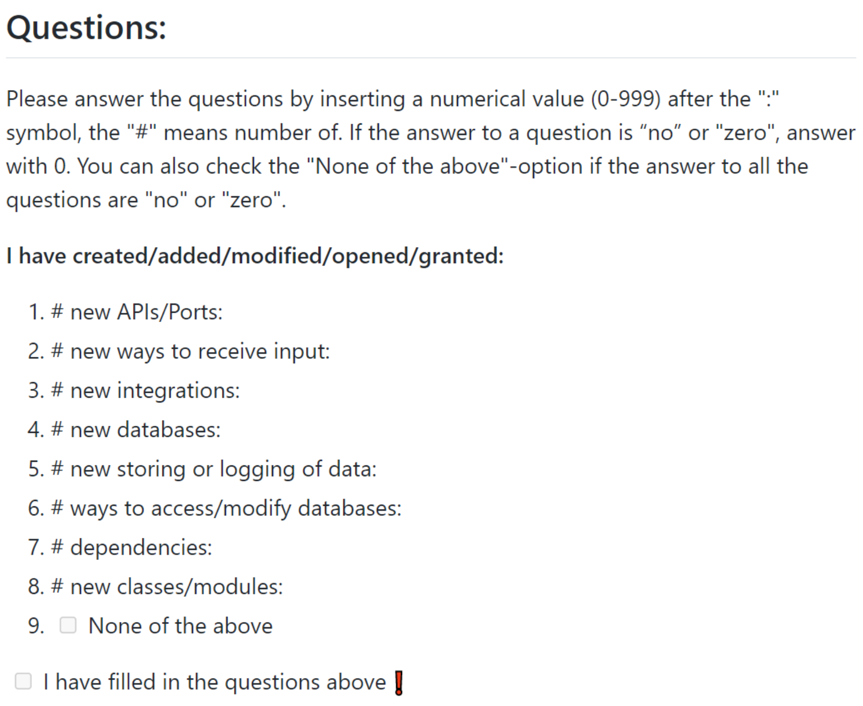
Appendix D

Implemented artifact output

This section contains the output of the implemented GitHub action artifact after the 18 workdays long case study at Visma. The questions that were used in shown in Figure D.1, and the results are illustrated in the subsections below. The results are discussed and explained in section 4.3.

D.1 The questions

The questions used for the case study at Visma is shown in the screenshot in Figure D.1.



Questions:

Please answer the questions by inserting a numerical value (0-999) after the ":" symbol, the "#" means number of. If the answer to a question is "no" or "zero", answer with 0. You can also check the "None of the above"-option if the answer to all the questions are "no" or "zero".

I have created/added/modified/opened/granted:

1. # new APIs/Ports:
2. # new ways to receive input:
3. # new integrations:
4. # new databases:
5. # new storing or logging of data:
6. # ways to access/modify databases:
7. # dependencies:
8. # new classes/modules:
9. ☐ None of the above

☐ I have filled in the questions above ❗

Figure D.1: Screenshot from GitHub of the form containing the questions that was used during the case study

D.2 Raw output

The following pdf contains the raw output of the GitHub implementation. The PR hashes has been anonymized and replaced with numbers, the same is the dates. This shows the answers from the participants for each question for each pull request.

Implemented GitHub anonymized raw output (Page 1 of 3)

Hash	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Date
1	0	0	0	0	0	0	1	0	1
2	1	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	2
5	0	0	0	0	0	0	0	0	2
6	0	0	0	0	0	0	0	0	2
7	0	0	0	0	0	0	0	0	2
8	0	0	0	0	0	0	0	0	2
9	0	0	0	0	0	0	0	3	3
10	0	0	0	0	0	0	0	0	3
11	0	0	0	0	0	0	0	1	3
12	0	0	0	0	0	0	0	0	3
13	0	0	0	0	0	0	0	1	4
14	0	0	0	0	0	0	0	0	4
15	0	0	0	0	0	0	0	0	4
16	0	0	0	0	0	0	0	0	4
17	0	0	0	0	0	0	0	0	4
18	0	0	0	0	0	0	1	0	5
19	0	0	0	0	0	0	0	0	5
20	0	0	0	0	0	0	0	0	5
21	0	0	0	0	0	0	1	0	5
22	0	0	0	0	0	0	0	0	5
23	0	0	0	0	0	0	0	0	5
24	0	0	0	0	0	0	0	1	5
25	0	0	0	0	0	0	0	0	5
26	0	0	0	0	0	1	0	0	5
27	0	0	0	0	0	0	0	0	6
28	0	0	0	0	0	0	0	0	6
29	0	0	0	0	0	0	0	0	6
30	0	0	0	0	0	0	0	0	6
31	0	0	0	0	0	0	0	0	6
32	0	0	0	0	0	0	0	0	6
33	0	0	0	0	0	0	0	0	7
34	1	0	0	0	0	0	0	1	7
35	1	0	0	0	0	0	0	0	7
36	0	0	0	0	0	0	0	0	8
37	0	0	0	0	0	0	0	0	8
38	0	0	0	0	0	0	0	0	8
39	1	0	0	0	0	0	0	0	8
40	0	0	0	0	0	0	0	0	8
41	0	0	0	0	0	0	0	0	9
42	0	0	0	0	0	0	0	0	9
43	0	0	0	0	0	0	0	0	9
44	0	0	0	0	0	0	0	0	9
45	1	0	0	0	0	1	0	0	9
46	0	0	0	0	0	0	0	0	9
47	0	0	0	0	0	0	0	0	9
48	0	0	0	0	0	0	0	0	9
49	0	0	0	0	0	0	1	0	9

50	0	0	0	0	2	0	0	8	10
51	0	0	0	0	0	0	0	0	10
52	0	0	0	0	0	0	0	0	11
53	0	0	0	0	0	0	0	4	11
54	0	0	0	0	0	0	0	0	11
55	0	0	0	0	0	0	0	0	11
56	0	0	0	0	0	0	0	0	11
57	0	0	0	0	0	0	0	0	12
58	0	0	0	0	0	0	0	0	12
59	0	0	0	0	0	0	0	9	12
60	0	0	0	0	0	0	0	0	12
61	0	0	0	0	0	0	0	0	12
62	0	0	0	0	0	0	0	0	12
63	0	0	0	0	0	0	0	0	13
64	0	0	0	0	0	0	0	1	13
65	0	0	0	0	0	0	0	0	13
66	0	0	0	0	0	0	0	0	13
67	0	0	0	0	0	0	0	0	13
68	0	0	0	0	0	0	0	0	13
69	0	0	0	0	0	0	0	2	13
70	0	0	0	0	0	0	0	0	14
71	0	0	0	0	0	1	0	0	14
72	3	0	0	0	0	0	0	0	14
73	0	0	0	0	0	0	0	0	14
74	1	0	0	0	0	0	0	0	14
75	1	0	0	0	0	0	0	0	14
76	0	0	0	0	0	0	0	1	14
77	0	0	0	0	0	0	0	0	15
78	0	0	0	0	0	0	0	0	15
79	0	0	0	0	0	0	0	0	15
80	0	0	0	0	0	0	0	0	15
81	0	0	0	0	0	0	0	0	15
82	0	0	0	0	0	0	0	0	15
83	0	0	0	0	0	0	0	0	15
84	0	0	0	0	1	0	0	5	15
85	0	0	0	0	0	0	0	0	15
86	0	0	0	0	0	0	0	0	16
87	0	0	0	0	0	0	0	0	16
88	0	0	0	0	0	0	0	0	16
89	0	0	0	0	0	0	0	0	16
90	0	0	0	0	0	0	0	0	16
91	0	0	0	0	0	0	0	0	16
92	0	0	0	0	0	0	0	0	16
93	0	0	0	0	0	0	0	0	16
94	0	0	0	0	0	0	0	1	17
95	0	0	0	0	0	0	0	4	17
96	0	0	0	0	0	0	0	0	17
97	0	0	0	0	0	0	0	0	17
98	0	0	0	0	0	0	0	0	17
99	0	0	0	0	0	0	0	0	17

Implemented GitHub anonymized raw output (Page 3 of 3)

100	0	0	0	0	0	0	0	0	17
101	0	0	0	0	0	0	0	0	17
102	0	1	0	0	0	0	0	0	18
103	0	0	0	0	0	0	0	0	18
104	0	0	0	0	0	0	0	0	18
105	0	0	0	0	0	0	0	0	18

D.3 The accumulated output without contribution factors

The following pdf containst the accumulated raw output of the GitHub implementation, without contribution factors.

Implemented GitHub anonymized accumulated output and accumulated contribution index with contribution factor)
(Page 1of 3)

Hash	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Date	Accumulated contribution index with factors
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	4	0	1	4
2	3	0	0	0	0	0	4	1	2	8
3	3	0	0	0	0	0	4	1	3	8
4	3	0	0	0	0	0	4	1	4	8
5	3	0	0	0	0	0	4	1	5	8
6	3	0	0	0	0	0	4	1	6	8
7	3	0	0	0	0	0	4	1	7	8
8	3	0	0	0	0	0	4	1	8	8
9	3	0	0	0	0	0	4	4	9	11
10	3	0	0	0	0	0	4	4	10	11
11	3	0	0	0	0	0	4	5	11	12
12	3	0	0	0	0	0	4	5	12	12
13	3	0	0	0	0	0	4	6	13	13
14	3	0	0	0	0	0	4	6	14	13
15	3	0	0	0	0	0	4	6	15	13
16	3	0	0	0	0	0	4	6	16	13
17	3	0	0	0	0	0	4	6	17	13
18	3	0	0	0	0	0	8	6	18	17
19	3	0	0	0	0	0	8	6	19	17
20	3	0	0	0	0	0	8	6	20	17
21	3	0	0	0	0	0	12	6	21	21
22	3	0	0	0	0	0	12	6	22	21
23	3	0	0	0	0	0	12	6	23	21
24	3	0	0	0	0	0	12	7	24	22
25	3	0	0	0	0	0	12	7	25	22
26	3	0	0	0	0	6	12	7	26	28
27	3	0	0	0	0	6	12	7	27	28
28	3	0	0	0	0	6	12	7	28	28
29	3	0	0	0	0	6	12	7	29	28
30	3	0	0	0	0	6	12	7	30	28
31	3	0	0	0	0	6	12	7	31	28
32	3	0	0	0	0	6	12	7	32	28
33	3	0	0	0	0	6	12	7	33	28
34	6	0	0	0	0	6	12	8	34	32
35	9	0	0	0	0	6	12	8	35	35
36	9	0	0	0	0	6	12	8	36	35
37	9	0	0	0	0	6	12	8	37	35
38	9	0	0	0	0	6	12	8	38	35
39	12	0	0	0	0	6	12	8	39	38
40	12	0	0	0	0	6	12	8	40	38

Implemented GitHub anonymized accumulated output and accumulated contribution index with contribution factor)
(Page 2of 3)

41	12	0	0	0	0	6	12	8	41	38
42	12	0	0	0	0	6	12	8	42	38
43	12	0	0	0	0	6	12	8	43	38
44	12	0	0	0	0	6	12	8	44	38
45	15	0	0	0	0	12	12	8	45	47
46	15	0	0	0	0	12	12	8	46	47
47	15	0	0	0	0	12	12	8	47	47
48	15	0	0	0	0	12	12	8	48	47
49	15	0	0	0	0	12	16	8	49	51
50	15	0	0	0	6	12	16	16	50	65
51	15	0	0	0	6	12	16	16	51	65
52	15	0	0	0	6	12	16	16	52	65
53	15	0	0	0	6	12	16	20	53	69
54	15	0	0	0	6	12	16	20	54	69
55	15	0	0	0	6	12	16	20	55	69
56	15	0	0	0	6	12	16	20	56	69
57	15	0	0	0	6	12	16	20	57	69
58	15	0	0	0	6	12	16	20	58	69
59	15	0	0	0	6	12	16	29	59	78
60	15	0	0	0	6	12	16	29	60	78
61	15	0	0	0	6	12	16	29	61	78
62	15	0	0	0	6	12	16	29	62	78
63	15	0	0	0	6	12	16	29	63	78
64	15	0	0	0	6	12	16	30	64	79
65	15	0	0	0	6	12	16	30	65	79
66	15	0	0	0	6	12	16	30	66	79
67	15	0	0	0	6	12	16	30	67	79
68	15	0	0	0	6	12	16	30	68	79
69	15	0	0	0	6	12	16	32	69	81
70	15	0	0	0	6	12	16	32	70	81
71	15	0	0	0	6	18	16	32	71	87
72	24	0	0	0	6	18	16	32	72	96
73	24	0	0	0	6	18	16	32	73	96
74	27	0	0	0	6	18	16	32	74	99
75	30	0	0	0	6	18	16	32	75	102
76	30	0	0	0	6	18	16	33	76	103
77	30	0	0	0	6	18	16	33	77	103
78	30	0	0	0	6	18	16	33	78	103
79	30	0	0	0	6	18	16	33	79	103
80	30	0	0	0	6	18	16	33	80	103
81	30	0	0	0	6	18	16	33	81	103
82	30	0	0	0	6	18	16	33	82	103
83	30	0	0	0	6	18	16	33	83	103
84	30	0	0	0	9	18	16	38	84	111
85	30	0	0	0	9	18	16	38	85	111
86	30	0	0	0	9	18	16	38	86	111
87	30	0	0	0	9	18	16	38	87	111
88	30	0	0	0	9	18	16	38	88	111
89	30	0	0	0	9	18	16	38	89	111
90	30	0	0	0	9	18	16	38	90	111

Implemented GitHub anonymized accumulated output and accumulated contribution index with contribution factor)
(Page 3of 3)

91	30	0	0	0	9	18	16	38	91	111
92	30	0	0	0	9	18	16	38	92	111
93	30	0	0	0	9	18	16	38	93	111
94	30	0	0	0	9	18	16	39	94	112
95	30	0	0	0	9	18	16	43	95	116
96	30	0	0	0	9	18	16	43	96	116
97	30	0	0	0	9	18	16	43	97	116
98	30	0	0	0	9	18	16	43	98	116
99	30	0	0	0	9	18	16	43	99	116
100	30	0	0	0	9	18	16	43	100	116
101	30	0	0	0	9	18	16	43	101	116
102	30	4	0	0	9	18	16	43	102	120
103	30	4	0	0	9	18	16	43	103	120
104	30	4	0	0	9	18	16	43	104	120
105	30	4	0	0	9	18	16	43	105	120

D.4 The contribution factors

The following table, Table D.1, is the factors used to calculate the accumulated contribution index with factors. The factors are just suggestions from the researchers, to indicate how factors will alter and affect the accumulated contribution factors.

Table D.1: The contribution factors used to calculate the accumulated contribution index.

Hash	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Risk factors	3	4	6	10	3	6	4	1

D.5 The accumulated output with contribution factors

The following pdf containst the accumulated raw output of the GitHub implementation, with contribution factors. Or in other words, the accumulated contribution index.

Implemented GitHub anonymized accumulated output and accumulated contribution index with contribution factor)
(Page 1of 3)

Hash	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Date	Accumulated contribution index with factors
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	4	0	1	4
2	3	0	0	0	0	0	4	1	2	8
3	3	0	0	0	0	0	4	1	3	8
4	3	0	0	0	0	0	4	1	4	8
5	3	0	0	0	0	0	4	1	5	8
6	3	0	0	0	0	0	4	1	6	8
7	3	0	0	0	0	0	4	1	7	8
8	3	0	0	0	0	0	4	1	8	8
9	3	0	0	0	0	0	4	4	9	11
10	3	0	0	0	0	0	4	4	10	11
11	3	0	0	0	0	0	4	5	11	12
12	3	0	0	0	0	0	4	5	12	12
13	3	0	0	0	0	0	4	6	13	13
14	3	0	0	0	0	0	4	6	14	13
15	3	0	0	0	0	0	4	6	15	13
16	3	0	0	0	0	0	4	6	16	13
17	3	0	0	0	0	0	4	6	17	13
18	3	0	0	0	0	0	8	6	18	17
19	3	0	0	0	0	0	8	6	19	17
20	3	0	0	0	0	0	8	6	20	17
21	3	0	0	0	0	0	12	6	21	21
22	3	0	0	0	0	0	12	6	22	21
23	3	0	0	0	0	0	12	6	23	21
24	3	0	0	0	0	0	12	7	24	22
25	3	0	0	0	0	0	12	7	25	22
26	3	0	0	0	0	6	12	7	26	28
27	3	0	0	0	0	6	12	7	27	28
28	3	0	0	0	0	6	12	7	28	28
29	3	0	0	0	0	6	12	7	29	28
30	3	0	0	0	0	6	12	7	30	28
31	3	0	0	0	0	6	12	7	31	28
32	3	0	0	0	0	6	12	7	32	28
33	3	0	0	0	0	6	12	7	33	28
34	6	0	0	0	0	6	12	8	34	32
35	9	0	0	0	0	6	12	8	35	35
36	9	0	0	0	0	6	12	8	36	35
37	9	0	0	0	0	6	12	8	37	35
38	9	0	0	0	0	6	12	8	38	35
39	12	0	0	0	0	6	12	8	39	38
40	12	0	0	0	0	6	12	8	40	38

Implemented GitHub anonymized accumulated output and accumulated contribution index with contribution factor)
(Page 2of 3)

41	12	0	0	0	0	6	12	8	41	38
42	12	0	0	0	0	6	12	8	42	38
43	12	0	0	0	0	6	12	8	43	38
44	12	0	0	0	0	6	12	8	44	38
45	15	0	0	0	0	12	12	8	45	47
46	15	0	0	0	0	12	12	8	46	47
47	15	0	0	0	0	12	12	8	47	47
48	15	0	0	0	0	12	12	8	48	47
49	15	0	0	0	0	12	16	8	49	51
50	15	0	0	0	6	12	16	16	50	65
51	15	0	0	0	6	12	16	16	51	65
52	15	0	0	0	6	12	16	16	52	65
53	15	0	0	0	6	12	16	20	53	69
54	15	0	0	0	6	12	16	20	54	69
55	15	0	0	0	6	12	16	20	55	69
56	15	0	0	0	6	12	16	20	56	69
57	15	0	0	0	6	12	16	20	57	69
58	15	0	0	0	6	12	16	20	58	69
59	15	0	0	0	6	12	16	29	59	78
60	15	0	0	0	6	12	16	29	60	78
61	15	0	0	0	6	12	16	29	61	78
62	15	0	0	0	6	12	16	29	62	78
63	15	0	0	0	6	12	16	29	63	78
64	15	0	0	0	6	12	16	30	64	79
65	15	0	0	0	6	12	16	30	65	79
66	15	0	0	0	6	12	16	30	66	79
67	15	0	0	0	6	12	16	30	67	79
68	15	0	0	0	6	12	16	30	68	79
69	15	0	0	0	6	12	16	32	69	81
70	15	0	0	0	6	12	16	32	70	81
71	15	0	0	0	6	18	16	32	71	87
72	24	0	0	0	6	18	16	32	72	96
73	24	0	0	0	6	18	16	32	73	96
74	27	0	0	0	6	18	16	32	74	99
75	30	0	0	0	6	18	16	32	75	102
76	30	0	0	0	6	18	16	33	76	103
77	30	0	0	0	6	18	16	33	77	103
78	30	0	0	0	6	18	16	33	78	103
79	30	0	0	0	6	18	16	33	79	103
80	30	0	0	0	6	18	16	33	80	103
81	30	0	0	0	6	18	16	33	81	103
82	30	0	0	0	6	18	16	33	82	103
83	30	0	0	0	6	18	16	33	83	103
84	30	0	0	0	9	18	16	38	84	111
85	30	0	0	0	9	18	16	38	85	111
86	30	0	0	0	9	18	16	38	86	111
87	30	0	0	0	9	18	16	38	87	111
88	30	0	0	0	9	18	16	38	88	111
89	30	0	0	0	9	18	16	38	89	111
90	30	0	0	0	9	18	16	38	90	111

Implemented GitHub anonymized accumulated output and accumulated contribution index with contribution factor)
(Page 3of 3)

91	30	0	0	0	9	18	16	38	91	111
92	30	0	0	0	9	18	16	38	92	111
93	30	0	0	0	9	18	16	38	93	111
94	30	0	0	0	9	18	16	39	94	112
95	30	0	0	0	9	18	16	43	95	116
96	30	0	0	0	9	18	16	43	96	116
97	30	0	0	0	9	18	16	43	97	116
98	30	0	0	0	9	18	16	43	98	116
99	30	0	0	0	9	18	16	43	99	116
100	30	0	0	0	9	18	16	43	100	116
101	30	0	0	0	9	18	16	43	101	116
102	30	4	0	0	9	18	16	43	102	120
103	30	4	0	0	9	18	16	43	103	120
104	30	4	0	0	9	18	16	43	104	120
105	30	4	0	0	9	18	16	43	105	120

Kristoffer Håkon Håkonsen

Triggering threat modeling in agile development

NTNU

Norwegian University of Science and Technology