Influencing the security prioritisation of an agile software development project

Inger Anne Tøndel, Daniela Soares Cruzes, Martin Gilje Jaatun, Guttorm Sindre

Abstract

Software security is a complex topic, and for development projects it can be challenging to assess what security is necessary and cost-effective. Agile Software Development (ASD) values self-management. Thus, teams and their Product Owners are expected to also manage software security prioritisation. In this paper we build on the notion that security experts who want to influence the priority given to security in ASD need to do this through interactions and support for teams rather than prescribing certain activities or priorities. But to do this effectively, there is a need to understand what hinders and supports teams in prioritising security. Based on a longitudinal case study, this article offers insight into the strategy used by one security professional in an SME to influence the priority of security in software development projects in the company. The main result is a model of influences on security prioritisation that can assist in understanding what supports or hinders the prioritisation of security in ASD, thus providing recommendations for security professionals. Two alternative strategies are outlined for software security in ASD – centralised and distributed – where we hypothesise that a distributed approach can be more relevant for SMEs doing ASD, and that this can impact how such companies should consider software security maturity.

1 Introduction

Today, software is an integrated and important part of daily life, as well as of our critical infrastructures, and it is essential that software has adequate security. What "adequate security" means is however unclear and may vary between different types of software projects and even with time as development progresses and requirements are negotiated (Tøndel et al. 2020a). Furthermore, which practices would be good to adopt to achieve this adequate level of security can depend on the development company and their development approach. Thus, software development projects need to make priorities and decisions related to security throughout development.

The challenge of prioritising security is present both in Agile Software Development (ASD) (Beck et al. 2001) and in more traditional development approaches (Blaine and Cleland-Huang 2008). However, as ASD is central in conventional software development, there is currently a need to address this challenge within a context of ASD. From a security standpoint, many have expressed scepticism towards ASD (Türpe and Poller 2017), and challenges related to security and other non-functional or quality aspects in ASD are extensively documented in several systematic literature reviews (Inayat et al. 2015; Oueslati et al. 2015; Khaim et al. 2016; Alsaqaf et al. 2017; Behutiye et al. 2020; Jarzębowicz and Weichbroth 2021). Challenges for prioritising security include missing or implicit security requirements (Khaim et al. 2016; Behutiye et al. 2020), a lack of incentives for security in the early stages of development (Oueslati et al. 2015; Behutiye et al. 2020), and security not being a part of agile frameworks (Oueslati et al. 2015) – all this leading to a neglect of security (Inayat et al. 2015; Oueslati et al. 2015; Behutiye et al. 2020; Jarzębowicz and Weichbroth 2021). But ASD also brings positive aspects related to security priority, e.g., through supporting security requirements iterations

(Türpe 2017), and the incompatibility of security and ASD has been declared a myth (Rindell et al. 2017).

Popular agile approaches such as Scrum (Schwaber 2004) do not have roles or activities specific for security. As a response, several extensions to Scrum and other agile frameworks have been developed to integrate security into the process (Williams et al. 2010; Pohl and Hof 2015; Rindell et al. 2015; Koç and Aydos 2017; Baldassarre et al. 2021). However, Scrum does not aim to prescribe how to perform the development work (including software security) in detail. Rather, it is a management framework aimed to "create an environment where development teams can self-organize and take responsibility for their work while being managed to the extent necessary for a project to succeed" (Türpe and Poller 2017). ASD values "Individuals and interactions over processes and tools" and trusts skilled and motivated software teams to do their job well (Beck et al. 2001). Thus, in ASD, the challenge of getting security prioritised needs to be addressed through interactions and support for teams rather than by prescribing specific ways of doing software security and its prioritisation (Türpe and Poller 2017; Weir et al. 2020a). Consequently, there is need to understand what supports and hinders prioritisation of security.

Research has identified a frequent divide between software security and information security (van Wyk and McGraw 2005; Tøndel et al. 2020c). Thus, the involvement of security professionals may be less than optimal in many organisations (Ashenden and Lawrence 2016; Thomas et al. 2018; Palombo et al. 2020). In Scrum, Product Owners are responsible for prioritisation of the backlog. Thus, Product Owners are key actors to interact with for security professionals who want to influence the priority given to security requirements. However, Product Owners have previously been identified as a common hindrance for quality aspects such as security, e.g., due to lack of knowledge, heavy workload, or insufficient availability (Alsaqaf et al. 2017).

This article provides insight into the strategy used by one security professional to influence the priority of software security in software development projects in the company. Through a longitudinal case study, we address the following two research questions (RQs):

- RQ1: What influences the security prioritisation throughout an ASD project?
- RQ2: How can security professionals increase the attention key decision makers give to security in an ASD project?

As literature generally claim that security requirements tend to be neglected in ASD (Inayat et al. 2015; Oueslati et al. 2015; Behutiye et al. 2020; Jarzębowicz and Weichbroth 2021), we expect that support for security prioritisation is important for software companies in general. Still, we chose to study a company that can be characterised as a small and medium sized enterprise (SME). Research points to SMEs as having the largest potential for software security improvements (Weir et al. 2020a). As SMEs are less likely to have a strong security department and a software security program that ensures software security to be addressed throughout development, they are less likely to be considered mature when it comes to software security, e.g., according to the Building Security In Maturity Framework (BSIMM) (Migues et al. 2021). Still, we suggest that also SMEs can and should strive towards adequate security in their products but expect that they need other ways to structure and think about such security initiatives than larger enterprises. A lot of our software is developed by SMEs. To illustrate, in Norway most software companies are of small or medium size. Thus, it is important to increase knowledge on how to support SMEs in making software with adequate security.

This article makes contributions to both theory and practice. Based on this case study we develop a model of influences on security prioritisation, organised into five influence categories. Then we relate

these findings to the state of the art, to build confidence in the model. The model can aid future research on security engineering in ASD, providing a framework for understanding. It can also help practitioners, especially security professionals, in navigating opportunities and challenges when trying to improve the priority of security in their projects.

This article is structured as follows. In Section 2 we explain the background for the research design. In Section 3 we explain our research approach. In Section 4 we describe the findings related to the two research questions. In Section 5 we introduce related work and relate it to our findings, in Section 6 we discuss our contribution, and in Section 7 we discuss the threats to validity. Section 8 concludes the article.

2 Background

This section explains the background for the research design, focusing on two key aspects: the need for an exploratory and inductive approach, and the decision to study a security expert's initiatives to improve security prioritisation.

2.1 The need for exploring security prioritisation in Agile Software Development

"Security is not simply a set of features or a functional component to be added to a system" (Türpe 2017). According to McGraw, "Software Security is the practice of building software to be secure and to function properly under malicious attack" (McGraw 2006). This implies that although security features are frequently necessary in a software system, other features also need to be secure, lest they be exploited by malicious attackers. Put in another way, it will likely be obvious to developers that an authentication mechanism needs to be secure, as attackers would like to compromise or circumvent it for illicit access to a system. However, any part of the software that reads data not provided by the developer is a potential target of attack (buffer overflows, SQL injection, etc. etc.). The large number of potential activities (BSIMM has 122 activities in its BSIMM12 version (Migues et al. 2021)), checklists (e.g., as in the OWASP Application Security Verification Standard (van der Stock et al. 2021)), and vulnerability and attack patterns (e.g., as organised within the Common Weakness Enumeration (CWE)[1] and the Common Attack Pattern Enumeration and Classification (CAPEC)[2]) all illustrate the broadness and the extensiveness of the software security work.

In this article we consider the concept *security prioritisation* to include prioritisation among security requirements and activities, prioritisation of security vs. other aspects such as functionality, as well as the priority and attention given to security in the day-to-day work. Thus, security prioritisation is a broad term that can encompass many different activities. Looking at BSIMM, activities like *[SM1.2] Create evangelist role and perform internal marketing*, *[PT1.1] Use external penetration testers to find problems*, and *[CMVM1.2] Identify software defects found in operations monitoring and feed them back to development*, although very different, all are likely to influence security prioritisation through raising the profile of security work and draw attention to specific vulnerabilities. This comes in addition to activities aimed at identifying security requirements and prioritise them for development.

To our knowledge, there are no studies that examine the priority given to security throughout a development project. Such a study can complement existing literature, e.g., on challenges to security in ASD (Inayat et al. 2015; Oueslati et al. 2015; Khaim et al. 2016; Alsaqaf et al. 2017; Behutiye et al. 2020; Jarzębowicz and Weichbroth 2021). The potential influences on the security priority are however numerous. In addition to security being a broad concern that includes functional as well as

[1] https://cwe.mitre.org/
[2] https://capec.mitre.org/

non-functional aspects (Türpe 2017), it is characterised by dispersed responsibilities as such a broad concern cannot solely be the responsibility of clearly defined security roles (Kocksch et al. 2018). Security prioritisation thus involves a broad set of individuals and their daily choices and priorities. Whereas the broad range of challenges is well documented in literature, more knowledge is needed on which challenges apply in which situations. Studies have shown that challenges identified in one company are not generally applicable to other companies (Karhapää et al. 2021; Olsson et al. 2021). This, and the lack of a theory on what brings priority to security in an ASD project, made us decide on an exploratory research design.

2.2 The role of security experts in security prioritisation in Agile Software Development

Security experts are not usually involved in requirements prioritisation in ASD. Prioritisation criteria, decision makers, and prioritisation frequencies may vary between projects but, typically, the most important prioritisation criterion is business value, with size, effort, and cost estimations being important inputs as well. Project constraints like release dates, budget, and available resources are important influences (Bakalova et al. 2011). The project backlog contains the requirements for the project, of which a prioritised subset is to be implemented in the upcoming iteration. Although the Product Owners are typically responsible for prioritising the backlogs based on expected business value (Türpe and Poller 2017), developers are often influential in practice, providing advice and suggesting solutions. Changes in prioritisation can stem from external changes or learning experiences (Bakalova et al. 2011).

While requirements prioritisation in ASD usually not involves security experts, security expertise is often considered a prerequisite for working with security requirements (Daneva and Wang 2018). In ASD, the relation between security experts and development teams is not always optimal (Ashenden and Lawrence 2016; Thomas et al. 2018; Tøndel et al. 2020c). However, a study of the adoption of secure development tools identified that "if companies structure their security processes so that security teams and other developers often interact, developers are more likely to feel personally responsible for security" (Xiao et al. 2014). Chowdhury et al. (2020) suggested that all organisational units have a strong relationship with the security department as a way of dealing with the implications of time pressure on security. Ashenden and Lawrence (2016) found that "when a security process works well, it's often because the security practitioner has good soft skills." Others have pointed to the need for combining top-down and bottom-up approaches to security (Cruzes and Johansen 2021). Including security experts in the development team, e.g., through the Security Champion role, is another suggestion (Antukh 2017; van der Veer 2019; Palombo et al. 2020; Tøndel et al. 2020c; Jaatun and Cruzes 2021; Tuladhar et al. 2021).

Much of the existing work on the involvement of security experts in ASD has been centred on moving software security to the developers, e.g., as is done by Palombo et al. (2020) and Tuladhar et al. (2021). Both represent ethnographic studies, and show how security experts effectively can bring security to developers through co-creation (Palombo et al. 2020) and situated learning (Tuladhar et al. 2021). However, there are fewer studies on how to bring security to Project Managers and Product Owners. This is the case although neglect of quality requirements (including security) is a challenge commonly reported in ASD (Behutiye et al. 2020; Jarzębowicz and Weichbroth 2021), and although Product Owners have been found to be a common hindrance for security (Alsaqaf et al. 2017; Terpstra et al. 2017). Daneva and Wang (2018) suggested to redefine the role of Product Owners when it comes to security requirements and their prioritisation, e.g., by having a Product Owner and a security expert share ownership over the backlog. With this article we contribute with knowledge on how security professionals can bring security to Product Owners.

Although we study the practices of a security expert, we are in this study more interested in what makes the security expert's actions have (or not have) an effect than in identifying a new or improved method or coping strategy for security prioritisation. Existing literature documents that a broad set of practices can be involved in work with security requirements (see Terpstra et al. (2017) and Daneva and Wang (2018) for overviews of coping strategies for security requirements in ASD). Further, there are specific techniques that offer support for prioritisation of security requirements. A prominent example is Protection Poker (Williams et al. 2010), a collaborative risk-estimation game that identifies and ranks security risks related to the features to be implemented in the upcoming iteration. Another is the approach by Ionita et al. (2019) that suggest a way to integrate risk assessment with security requirements prioritisation to populate the product backlog with prioritised security requirements. But despite availability of such techniques, there is limited knowledge on what coping strategies are most beneficial and why (Tøndel and Jaatun 2020), and there is still the challenge of being able to make security gain priority in practice (Türpe and Poller 2017). This work contributes with knowledge on what supports and hinders prioritisation of security.

3 Research approach

Our research approach is exploratory. Modern software development can be complex or messy in nature (Pelrine 2011; Tøndel et al. 2020b) and thus, a plethora of potential influences exist. Case studies are suited for in-depth investigation of complex contemporary phenomena where the boundary between context and phenomenon can be unclear (Yin 2018). A longitudinal case study allowed for a proper attention to context as well as a thorough investigation of a broad set of influences.

3.1 The case

The research questions call for studying a case with changing security prioritisation and with security professionals working to increase security attention among key decision makers. Through a research project with several company participants, we had access to a case that matched these needs. The company – subsequently called DevCo – was an SME with about 80 developers across four locations. The main office of the company was in Norway and within reasonable travelling distance for the researchers. The other locations were in Eastern Europe and in two of the Nordic countries. They developed software solutions on a contract basis, but with the aim to, through these contracts, develop products that could be offered to a broader set of actors within the sector in which they operated. Solutions included mobile apps for the public, hardware-oriented solutions for real time monitoring, and back-end solutions. In DevCo, development was performed according to Scrum in the main aspects we wanted to investigate (development in iterations; backlog; Product Owner role responsible for requirements refinement and prioritisation; autonomous teams; etc.) although the environment of the project (e.g., the bid process and the contracts) was not fully agile. The only central security resource was a Security Officer in a 60 % position, and the Security Officer role was placed in the development department. DevCo had some experience on including software security in some previous projects and had recently increased their attention to security through hiring a Security Officer and establishing a Security Champion role in the development team. Still, DevCo lacked systematic attention to security on the Product Owner and Project Manager level.

At the time of the study, DevCo had just received a big new project – in the following called ProjectAlpha – where the customer had more explicit security requirements than what DevCo had experienced before, thus motivating the need to improve on software security. In ProjectAlpha they were to develop front-end solutions for Android and iOS, as well as back-end solutions that

integrated with security solutions from a third party. Not long after, they started a second big project – ProjectBeta – where there was less security push from the customer and where the technology was more complex, involving more hardware components. The Security Officer used the security push from the customer in ProjectAlpha to start a new initiative, in the following called the Security Requirements Initiative. This initiative offered a process to elicit, document, prioritise, and follow up on security requirements, and ProjectAlpha was its pilot.

For us, DevCo offered an opportunity not only to study the changing security priorities (RQ1) and the impact of security expert initiatives (RQ2). We also considered the case to have characteristics that we suspected to be common among SMEs – making this a relevant case to study in a single-case study design (Yin 2018) – at the same time as it was interesting from a theoretical standpoint. We assumed that for SMEs it would be common to have few dedicated security resources, to have some software security experience but without a strong software security program (thus limited software security maturity), and to have established ASD practices but with challenges to change the larger environment surrounding the company to make the full project agile. Still, the relatively short distance between security experts and development allowed them to interact regularly and thus study this interaction. Further, studying an SME made it more feasible to aim for an overview of a broad set of influences than what we expect would have been possible with a larger development company – taking into account the invisibility of security and security work (Kocksch et al. 2018).

A further benefit of the case was that two new projects were starting almost simultaneously, allowing us to opt for an embedded single-case design with each project as a single unit of analysis (Yin 2018). Thus, we could study similar interactions and initiatives in two projects and learn from similarities and differences that would be identified among them. However, our main emphasis was on ProjectAlpha as the Security Requirements Initiative pilot. In this project the key participants were positive to participating in this study and willing to act as interviewees etc., more so than in ProjectBeta. According to our embedded single-case design we considered each project as a unit of analysis, but we analysed data from ProjectAlpha first, using an inductive approach to identify influences on security priority and understand the implications of the Security Officer and the Security Requirements Initiative. Then we analysed data from ProjectBeta with a deductive approach, based on the findings from ProjectAlpha. This allowed us to study one project in detail (ProjectAlpha) while strengthening and extending upon the findings with additional data from another project (ProjectBeta) in the same company.

ProjectAlpha had a duration of around two years. It involved one team where most of the developers were in the Eastern Europe office, while the project management as well as the Security Champion and the Security Officer were in the main office in Norway. The project had two Product Owners. One of them had extensive understanding of the customer domain and was responsible for the front end. The other, sometimes in the following referred to as the technical Product Owner (TPO), had a strong technical competence and had previously worked as a developer/consultant at DevCo. In addition to Product Owner responsibilities for the back end, the TPO had responsibility for the architecture of the overall product. The Project Manager of ProjectAlpha was responsible for monitoring the contract activities and following up the customer, the budget, and the requirements. This Project Manager was new to DevCo, and ProjectAlpha was his first main project in DevCo. The developers (including the Security Champion) and the Security Officer were part of the development department, while the Project Manager and the two Product Owners were in the project management department. Operations, that were to be responsible for operation of the software solution after development, was organised in yet another department. ProjectBeta had a similar

organisation except that it consisted of more teams, and that the Security Champion and the Product Owner was placed at other office locations.

3.2 Data collection

In the case study, we used multiple methods of data collection. As can be seen from the overview given in Figure 1, data collection took place between April 2018 and June 2020 and included observations, interviews, status updates with the Security Officer, and documentation of processes and requirements. We opted for such varied data collection because we wanted to get a broad view of the security prioritisation in ProjectAlpha and get to a rich description of this case. Observations allowed us to observe key security discussions first-hand, status updates with the Security Officer and interviews allowed us to collected personal experiences from the main actors, and access to key documentation items such as security requirements allowed for direct knowledge about the projects' requirements and how they were documented throughout the project. Data collection was spread out in time throughout the whole period of the study, with an emphasis on observations in the beginning, interviews towards the first release of ProjectAlpha, and status updates with the Security Officer throughout. To properly account for the context, we did not limit data collection to the two projects but collected supplementary data from surrounding activities such as department and security guild meetings. We aimed to observe as many as possible of the scheduled meetings where security priorities were to be discussed in ProjectAlpha. For interviews, we recruited four key individuals: the Security Champion, the two Product Owners and the Project Manager. These were selected due to their involvement in the Security Requirements Initiative and their influence on project priorities. From ProjectBeta we did try to recruit both the Product Owner and the Security Champion, however we were only able to get an interview with the Security Champion. All data collection was done by the first author.

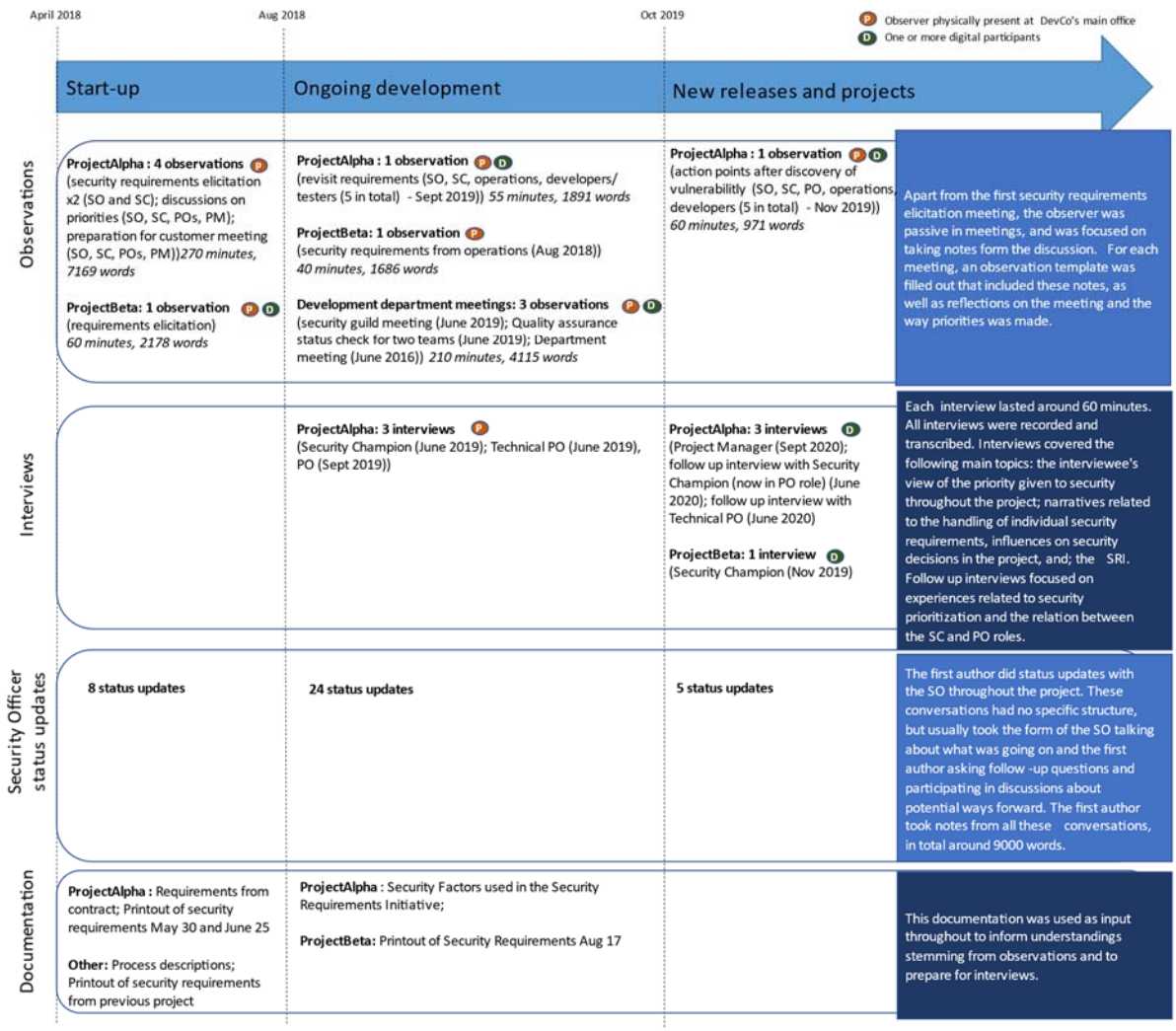

Figure 1. Overview of data collection (SO = Security Officer, SC = Security Champion, PM = Project Manager, PO = Product Owner, SRI = Security Requirements Initiative)

3.3 Analysis

We needed an analysis approach that could help us make good use of our multi-method data collection and support us in our need to take a broad view of the situation to identify influences on security priorities. We opted for an analysis approach based on thematic coding (Maxwell 2013), Situational Analysis (SA) (Clarke et al. 2016) and Narrative Analysis (NA) (Riessman 2008). Thematic coding supported us in categorizing and structuring the collected data into themes. SA and NA, on the other hand, helped us identify connections in the data material through analysing the situation as a whole (SA) and through analysing narratives in an unfragmented state (NA). Figure 2 gives an overview of the analysis process for arriving at the influence categories (RQ1).

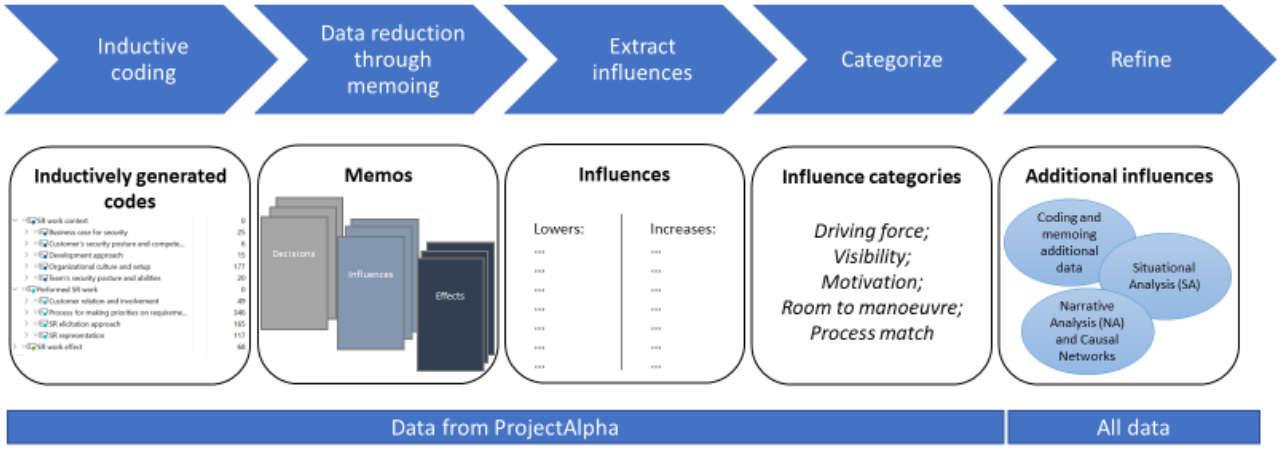

Figure 2. Analysis process for arriving at influences and influence categories

Coding of data was performed in the tool MAXQDA. Although the coding approach was inductive, we used categories from the conceptual model of Tøndel and Jaatun (2020) to organise the inductive codes within concepts that had already been identified as relevant based on previous studies: the context of the security work including the security posture of the customer, the organisation, and the team; the way the security work that was performed, and; its effect. This helped us get some initial structure to the data and increase overview from the onset.

As influences on security priorities can be related both to contextual aspects as well as the technique and the effect observed, influences on the priority given to software security could be found within all our organising codes. To capture the broad set of influences, and at the same time make analysis more manageable, we utilized memoing (Maxwell 2013). For each of the topics "security decisions", "influences", and "impact" we created a longer memo where we wrote a summary of all the codes that were related to the topic, and we used the functionality of MAXQDA to link relevant codes to the memos for traceability. We made three versions of all these memo types; one (I) based on data from interviews, and two (II and III) based on observations and Security Officer status updates – where II considered the start-up phase (ref. Figure 1) and III the rest of the project. This approach was taken to make the analysis more manageable, not taking the full data material into account at once, and to allow for identifying similarities and variations in findings among these data sources on key issues (Eisenhardt 1989). All research memos were discussed with the Security Officer of DevCo, and comments from the Security Officer to the memos were noted down. Generally, the Security Officer did not object to the findings in the memos, except for a few minor corrections, but sometimes had additional comments and further explanations of phenomena described in the memos.

From all these nine memos, we extracted influences that lowered and increased the priority given to security. This resulted in two long lists of influences, subsequently refined and categorised using the tool MindManager. In this process we compared the identified influences and grouped those that were similar into refined influences. Finally, we grouped the identified influences into five influence categories that emerged from this categorisation process (driving force, visibility, room to manoeuvre, motivation, and process match) and developed a definition for each of these categories. Figure 3 demonstrates how the influences we identified through this analysis process can be traced back to coded segments.

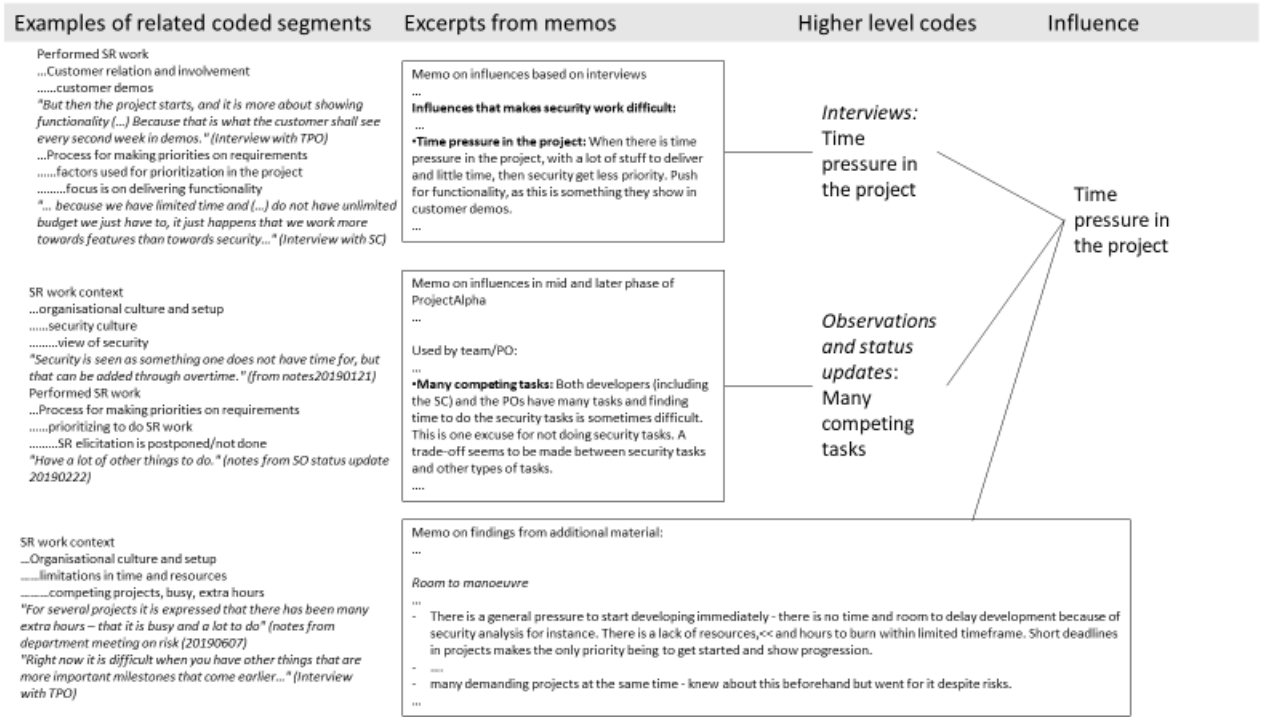

Figure 3. Example showing part of the link between one identified influence and coded segments, demonstrating how the identified influences are related to memos that are linked to coded segments from the data material that are ordered according to categories from (Tøndel and Jaatun 2020)

To strengthen the results from the study of ProjectAlpha, we analysed data from ProjectBeta and from the surrounding context. For this additional material, we performed deductive coding. We deliberately looked for data that could dispute our findings from ProjectAlpha or expand our understanding of the already identified influences.

SA and NA were utilised to build a stronger understanding about the identified influences and to reduce the risk of missing important influences in our analysis. Constructing a Social Worlds/Arenas Map (Clarke et al. 2016) for ProjectAlpha helped us get an overview of all the collective actors that were to a smaller or larger extent influencing the priority given to security in this project. Constructing Positional Maps (Clarke et al. 2016) helped us get an understanding of an important underlying debate in the data material, that of whether one should rely on software craftmanship and everybody taking responsibility for security, or whether there should be a central push for security, e.g., through procedural requirements and audits. NA supported us in analysing the narratives collected in interviews in an unfragmented manner, looking at the content and the flow of events. We also constructed our own narratives of the evolvement of the Security Requirements Initiative based on status updates with the Security Officer, and observations.

To visualise the flow of events in the collected and constructed narratives we created causal networks (Miles et al. 2018). These causal networks helped us understand how states and/or actions were understood as interrelated into sequences. The causal networks supported us in the refinement of the influences. However, their main importance was in their support for identifying and understanding strategies applied by the Security Officer and the underlying mechanism at play (RQ2).

4 Findings

Through the analysis we identified a large set of influences on the priority given to security, and we organised them into five categories (Figure 4): driving force, visibility, motivation, room to manoeuvre, and process match. These categories together cover aspects of the individuals, the project, and the company. In the following we provide a definition of each of these categories before the following subsections explain the influences observed within each of these categories:

- ***Driving force*** refers to someone who takes initiative and responsibility for making software security happen. A negative driving force would actively hinder software security.
- ***Visibility*** refers to the degree to which security is visible (seen, known about) to stakeholders related to the project. This includes the visibility of security to developers in their daily coding activities, to project management and top management, to the customer, and in the product.
- ***Motivation*** refers to the willingness to focus on software security, as well as the aspects that cause such willingness. Reasons for doing or not doing software security, and activities that provide such reason would be part of this category.
- ***Room to manoeuvre*** refer to resources and opportunities to prioritise software security, and to act accordingly. This might include time, budget, competence, etc.
- ***Process match*** refers to the ability to fit the security approach into the existing software development process, so that they align well.

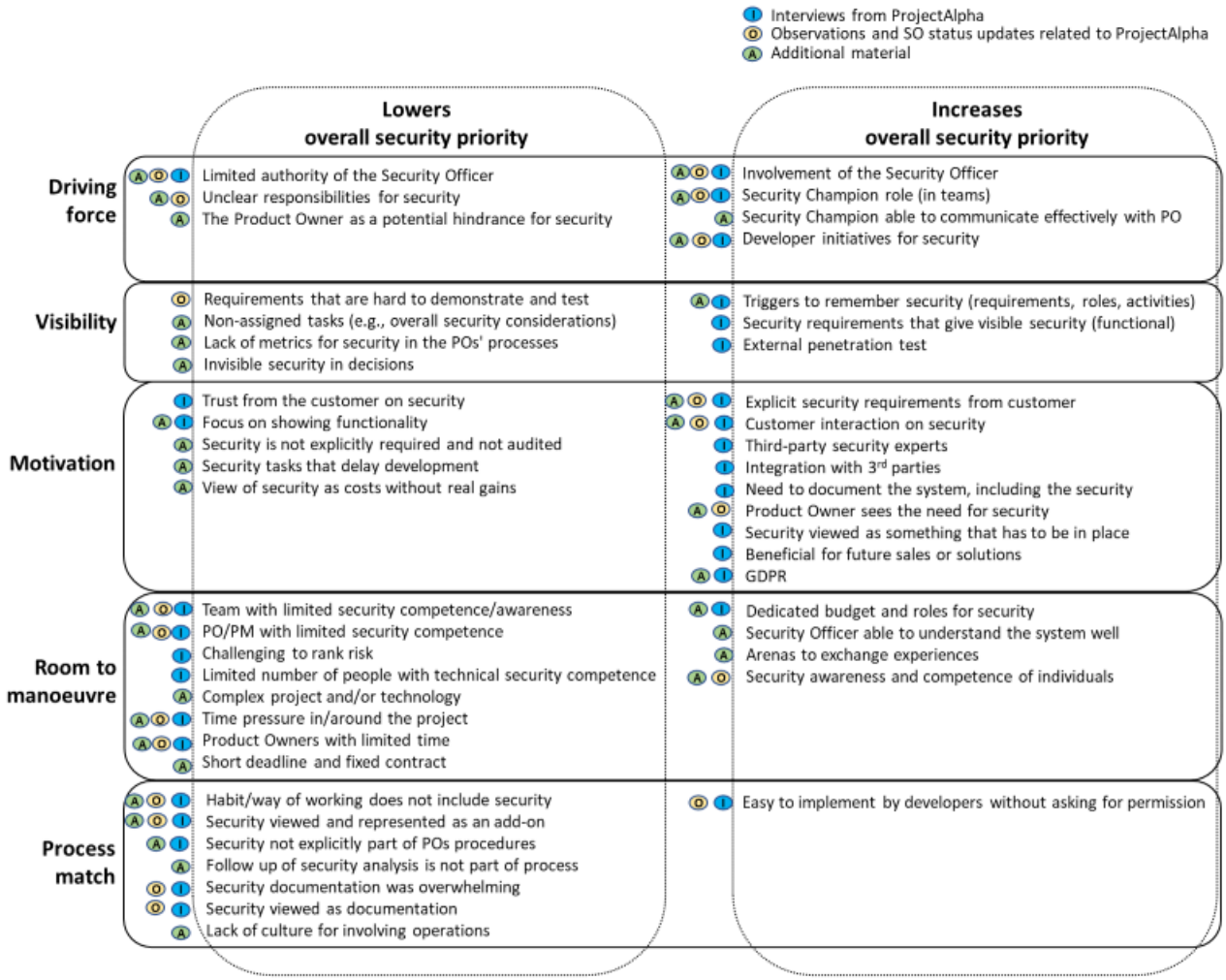

Figure 4. Conceptual model of influences on security priority (PM = Project Manager, PO = Product Owner)

Note that we can see in the data that these five influence categories are somewhat related, and we found that some of the influences could fit more than one category. In such cases, we chose the category that was most explicitly linked to the influence. Visibility plays an important role for motivation, and it was sometime hard to distinguish between the two, but we still opted for keeping both categories as motivation can happen also without visibility. Room to manoeuvre and process match can inhibit motivation for security, e.g., in cases where these pose limitations for security that are hard to overcome. Driving force is related to motivation and room to manoeuvre, as individuals with a motivation for security and room to take on security tasks are more likely to take on responsibility and push for security. Driving force is also related to visibility, as a driving force can contribute to security being more visible, and it is related to process match as, e.g., organisational structures can strengthen or limit the potential influence of an individual or a role.

In the following we explain the how the identified influences played out in the study, as well as the strategies used by the Security Officer related to these influence categories. Figure 5 gives an overview of the evolution of the Security Requirements Initiative that the Security Officer initiated.

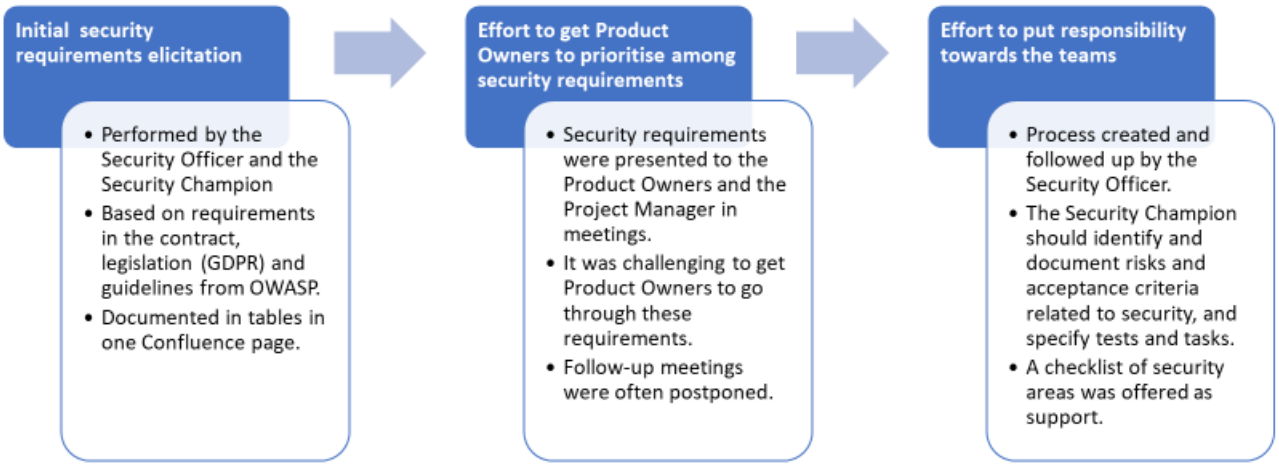

Figure 5. Overview of the stages of the Security Requirements Initiative

4.1 Driving force

There were several roles that had a potential influence on the priority given to security in this project. Figure 6 gives an overview of these roles, through a social worlds/arenas map created through Situational Analysis (Clarke et al. 2016). In this figure, the size of the oval shapes and their overlap with the main concern (the priority given to security in ProjectAlpha) represent our understanding of the magnitude of their influence. Table 1 provide a description of their influence. As can be seen from this figure and table, the main driving forces for software security were the Security Officer and the Security Champion.

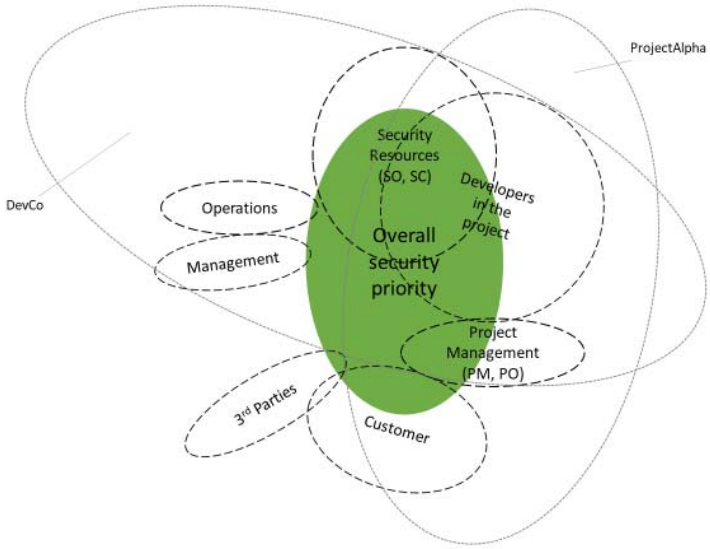

Figure 6. Social worlds/arenas map for ProjectAlpha (SO = Security Officer, SC = Security Champion, PM = Project Manager, PO = Product Owner)

Table 1. Overview of the social worlds/arenas involved and their influence as Driving Force

Social world/arena	How they influenced the priority given to security (including the battling of challenges)	Influence in conceptual model (Figure 4)
Security Resources: Security Officer	• Initiated and followed up the Security Requirements Initiative, arranged meetings and poked individuals about security tasks. • Later, the Security Officer left the company and a drop in security focus was observed. • Had to battle issues with formal authority, as the Product Owners, the Project Manager, and operations were not in the development department. • It was not scalable to be personally involved in all follow-ups on security.	+ Involvement of the Security Officer - Limited authority of the Security Officer
Security Resources: Security Champion	• Supported continuity of security attention in the development teams. • More successful when able to present security issues to Product Owners in a way that made them understand the costs and the risks and made them able to make an informed decision.	+ Security Champion role in teams + Security Champion able to communicate effectively with Product Owner
Developers in the project	• Several developer initiatives for security came in addition to the explicit security requirements from the customer. • Developers did the coding and made choices on which security mechanisms to apply. • Influence on the project management roles because of technical competence.	+ Developer initiatives for security
Project Management (Project Manager, Product Owners)	• Influence through prioritisation and budget, but largely delegated prioritisation to the security resources, and to some extent to the developers. • Product Owners explained that they relied on the ability of the Security Champion, developers, or the Security Officer to bring important security issues to their attention. • Viewed as more of a hindrance for security in ProjectBeta, as the Security Champion felt their security concerns were not taken seriously by the Product Owner.	- Unclear responsibilities for security - The Product Owner as a potential hindrance for security
Customer	• Indirect influence through the explicit security requirements. • Still, it seems security was not that visible in communication with the customer throughout, except from occasions such as presentation of security analysis or discussions concerning specific security requirements.	(Indirect influence, thus not included as driving force)

3rd Parties (vendors of software components they had to integrate with; security experts involved by the customer)	• Technological solutions from 3rd parties could support (or not support) the security requirements, thus indirectly allowing for (or hampering) their prioritisation. • 3rd parties involved by the customer were perceived as security experts and influenced through the security competence they brought and the push they represented toward security.	(Indirect influence, thus not included as driving force)
Management	• Management influence was perceived differently by the individuals at DevCo; some perceived a support from senior management on spending resources on security, while others explained that senior management expected them to do security but did not understand that this had a cost.	(No clear examples of how this influence played out, thus not included as driving force)
Operations	• Had security competence that could have benefited development and knew how security was addressed during operation, but the silo structure limited operation's involvement during development. There were ongoing initiatives to improve that.	(Potential driving force, but not much active in the project)

For the Security Officer, it was highly useful to have the Security Champion role as a driving force for security within the development team. The Security Officer collaborated closely with the Security Champion both in performing the activities of the Security Requirements Initiative and in increasing its adoption. At the Product Owner and Project Manager level, however, there was no similar champion role to collaborate with, and the Security Officer had challenges in being the driving force for security towards Product Owners due to the organisation of the Security Officer role in the development department. The Security Officer's interaction with operations was challenging for similar reasons; when the Security Officer made efforts to involve operations in ProjectBeta, the Security Officer got the perception that operations viewed interaction with the Security Officer as doing development a favour.

Aspects related to trust and view of responsibility among individuals and roles influenced the adoption of the Security Requirements Initiative in ways that were not always easy to predict for the Security Officer. When the Project Manager and the less technical Product Owner saw that security was addressed by someone else, this resulted in a less perceived need to take active responsibility for security themselves. Moreover, when the Security Officer arranged security onboarding meetings with developers, they got the impression that they had to do security but no one else had to, and they pushed back on this message.

4.2 Visibility

Both the priority given to security and the visibility of security varied throughout the project in a way that indicates that they are linked. Figure 7 shows a generalised view of how the priority given to security was depicted in interviews. As shown, it varied in a U-curve but with some spikes along the way. The drop in the U-curve was explained in the following way in one of the interviews (shortened and paraphrased): *We had quite high attention to security in the beginning, as part of planning. And we started development with an aim to do security well and document it well. But then, towards the*

middle, some of this had been, maybe not forgotten but at least not as prioritised. The focus was on making sure to finish, deliver, and make money on the project.

This "forgetting" or down-prioritising of security can be explained partly by limited visibility of security, as described in Table 2. Various triggers for security were important for the spikes, including triggers related to requirements, security roles and activities such as pentests. This made security visible and on the agenda of both the Project Manager and the Product Owner roles. On the other hand, a lack of visibility of security in formal routines lowered the attention given to security.

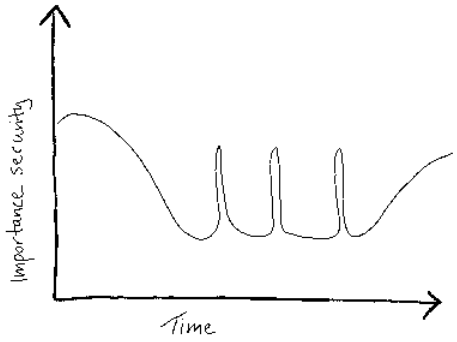

Figure 7. Typical timeline drawn in interviews.

Table 2. Overview of the role of influences related to visibility on the security priority timeline in Figure 7

Stage in timeline	**How this was influenced by visibility**	**Influence in conceptual model (Figure 4)**
Initial high security attention	• Security analysis was required from the customer in the beginning of the project. • The Security Requirements Initiative pushed for the security priorities being made early in the project.	+ Triggers to remember security

Drop in security attention	• The security requirements were less visible than the functional requirements, and this was particularly the case for security requirements that were more overarching or unclear, and thus hard to demonstrate and test. These requirements were often postponed. • A lack of security focus in the Product Owners' processes (e.g., in form of requirements for security analysis or Key Performance Indicators (KPIs) on security) resulted in security work not being visible and accounted for. One Product Owner compared security to testing and explained that testing was part of the procedures and processes and were part of KPIs, and thus testing was done despite time pressure and although the management and Product Owner level often thought that testing took too much time. • The security work that was done often ended up being quite invisible as well (e.g., only known about by developers), and thus did not help build security culture in the same way as if it would have been more visible.	- Requirements that are hard to demonstrate and test - Non-assigned tasks - Lack of metrics for security in the Product Owners' processes - Invisible security in decisions
Some ongoing attention, despite the drop	• The Security Officer and the Security Champion roles served as visual reminders of security. This could take the form of remembering that there were security tasks to be done when seeing the Security Officer in the office, or in case of the Security Champion: *"It becomes kind of, we can call it security advertisement. You can see that he has to spend hours on security champion work"* (as explained by one of the Product Owners).	+ Triggers to remember security
Spikes	• In iterations where security requirements were to be implemented, security was more visible. • Security requirements that gave visible security (were functional) served more as a trigger for security and were easier to give priority, as opposed to more overarching security requirements. • Customer interaction on security (e.g., presentations, questions) put security on the agenda of Project Managers and Product Owners. • Performing an external penetration test increased visibility of security towards the Project Manager and Product Owner roles.	+ Triggers to remember security + Security requirements that give visible security + External penetration test
Increase towards release	• Security became visible as part of a need to check whether all the explicit security requirements from the contract were fulfilled.	+ Triggers to remember security

The Security Officer was active in increasing the visibility of security throughout the project, and actively made use of security triggers by arranging security meetings, pushing for security through

documentation, and poking about security tasks. Note however that the Security Officer experienced a need to strike a balance between reminding key individuals about their security tasks and respecting that they were pressed for time.

4.3 Motivation

The drops as well as the spikes in security attention can also to some extent be explained by motivational factors. This is outlined in Table 3. Motivation and view of security varied from team to team, and between individuals. In ProjectAlpha, the interaction and relation to the customer was important for the motivation for security, as illustrated by the following narrative from an interview (shortened and paraphrased): *We were bound by the requirement from the customer that there should be a security analysis. Thus, we worked on that to get a good start. And then, at some point, we presented this to the customer and their technical consultant. And we received very good feedback on the work we had done, they were impressed, had never had software vendors that had that much security focus. And then we got this drop because we got a little complacent.* ProjectAlpha additionally benefited from key individuals with a positive view of security, as expressed in the following interview quote: *"It is a huge difference! In previous projects it has not been, let's say, a first-level thing. It has been delivering of features and ensuring customer requirements and then security is some murky stuff in the bottom that we should take care of if time. But now it has become something that has to be in place."* In ProjectBeta, on the other hand, there were key individuals which did not seem to be as positive towards security.

Table 3. Overview of the role of influences related to motivation on the security priority timeline in Figure 7 – note that as ProjectAlpha and ProjectBeta experienced main differences when it comes to security motivation, Table 3 represent findings from ProjectAlpha unless otherwise specified.

Stage in timeline	**How this was influenced by motivation**	**Influence in conceptual model (Figure 4)**
Initial high security attention	• Having to go through the explicit security requirements from the customer increased understanding on what to deliver on security and increased motivation to do a good job on security. • Presentation of the security analysis to the customer led to a motivation to do a good job on this analysis.	+ Explicit security requirements from customer + Customer interaction on security
Drop in security attention	• They experienced perceived trust from the customer on security. Throughout, customer meetings emphasised the showing of functionality. • The motivation for security was reduced when security was not explicitly required in procedures or in requirements and were not audited. This especially became a motivational challenge in combination with security tasks that delayed development. • In ProjectBeta, the Security Champion experienced a Product Owner with a view of security as cost without real gains. Thus, the motivation to spend time on upgrading the	- Trust from the customer on security - Focus on showing functionality - Security is not explicitly required and not audited - Security tasks that delay development - View of security as costs without real gains

	security of an already working solution was low, and the approach to security became more reactive.	
Some ongoing attention, despite the drop	• Security was viewed as something that had to be in place and was discussed as something that could be beneficial for future sales and solutions. • Security experts were in the loop on the customer side and the project had to interact with these experts: *"Of course, when you have good people on the other side then you want to deliver good work too. (…) And I use them, (…) ask questions."* • The need for integration with solutions from 3rd parties, and thus the need to secure that interaction, increased the perceived need for and the motivation to handle security. • GDPR represented a push and motivation for security, and requirements related to GDPR came up in the observed meetings. • Security aware Product Owners knew that the customer needed security, even if it was not explicitly stated or if it was not pushed for in the same way as features. However, it could be challenging for Product Owners to balance what the customer explicitly stated as requirements with what the Product Owner knew the customer needed to have in addition to that (e.g., security).	+ Third-party security experts + Integration with 3rd parties + Product Owner sees the need for security + Security viewed as something that has to be in place + Beneficial for future sales or solutions + GDPR
Spikes	• Customer interaction on security (e.g., presentations, questions) increased the motivation to do a good job on security.	+ Explicit security requirements from customer + Customer interaction on security
Increase towards release	• There was a need to check that all the promised security had been delivered. Meetings were arranged to discuss this. • The need to document the system, including the security, could motivate for and put attention on security. Such documentation could, e.g., be part of release notes for customers or operations.	+ Explicit security requirements from customer + Need to document the system, including the security

One of the strategies used by the Security Officer, was to put attention on the explicit security requirements coming from the customer. Further, the Security Requirements Initiative itself pushed for security through creating a process for security requirements elicitation, documentation and follow up, and thus was an aim to increase motivation for security. Despite several challenges related to its adoption, we observed strong positive effects related to increased security awareness. Interviewees stated that, to some extent, this awareness spread also to other roles in the project, including testers and developers. Even meetings that seemed to be quite unsuccessful led to improved security awareness. It is likely that this increased security awareness had broader

consequences in form of more developer initiatives for security and less pushback from Product Owners, etc., though we have no clear data to support this causality.

4.4 Room to manoeuvre

The room to manoeuvre is dependent on resources such as time, budget, and competence. Table 4 shows how these resources influenced the priority given to security. Of particular importance for getting ongoing priority were the dedicated security roles that represented time and budget for security, as well as competence. On the other hand, the strong time pressure experienced – especially by Product Owners – represented a major hindrance for giving priority to security.

Table 4. Overview of the role of influences related to room to manoeuvre

Resource	**How this influenced the priority given to security**	**Influence in conceptual model (Figure 4)**
Time and budget	• The Security Champion and Security Officer roles had pre-allocated time to work on security tasks. • Product Owners already had a high workload with limited possibility to take on more tasks. It was challenging for them to find time to work on the security requirements. • Product Owners explained that short deadlines led to a strong pressure to start development immediately but, at the same time, because of the fixed contracts it was essential to get security in before development started. Thus, security ended up requiring a delay in development that there was no room for. • Short deadlines put a lot of pressure on Product Owners who were often involved in several projects at the same time, all at different stages. Thus, when the Security Champion later became a Product Owner he found himself *"surprised that I am not performing better than those that were* Product Owners *while I was Security Champion and developer".*	+ Dedicated budget and roles for security - Time pressure in/around the project - Product Owner with limited time - Short deadline and fixed contract
Competence	• The influence of the Security Officer was dependent on the Security Officer's ability to understand the technology of the projects, and thus was stronger in ProjectAlpha than in ProjectBeta where the Security Officer was less able to understand the system under development. • Both the Security Officer and the Security Champion roles contributed to the availability of arenas to exchange experiences between projects on security. This happened through the Security Guild that allowed for discussions among all Security Champions and through the Security Officer being involved in a broad set of projects.	+ Security Officer able to understand the system well + Arenas to exchange experiences + Security awareness and competence of individuals - Team with limited security competence/ awareness

	• The security in the development projects was highly dependent on the security awareness and competence of the developers, as they were creating their own security initiatives. Security competence and awareness varied greatly among the development teams of DevCo, and among Product Owners. • Security competence is important to perform security activities well. ProjectAlpha experienced that in meetings it was challenging to rank risk, a task that required broad security competence, and it could be difficult to know which security requirements were most important. • With few individuals with security competence, these became a bottleneck. • More complex projects put stronger demands on competence. Security seemed to suffer in ProjectBeta which was more complex both in terms of technology and team structure.	- Product Owner / Project Manager with limited security competence - Challenging to rank risk - Limited number of people with technical security competence - Complex project and/or technology

The Security Officer experienced that time pressure led to a resistance to take on security tasks, and used several strategies to address this resistance (focusing on security requirements from the customer, splitting into smaller tasks, pushing security through documentation and meetings, and poking about security tasks). Progress was difficult as it relied on Product Owners who did not have security as part of their procedures and did not have time for additional tasks. This issue was difficult for the Security Officer to address, as it depended on the overall project load and the contracts with the customer. Thus, adding security to the beginning of the project was not early enough, one needed to also consider the bid process (and this was now no longer an option). To further emphasise this need, it turned out that some of the security requirements stemming from the bid process of ProjectAlpha did not make much sense and were costly to address, and thus had to be renegotiated.

Regarding competence, the Security Officer supported learning between projects. Further, the activities and meetings initiated as part of the Security Requirements Initiative increased security awareness and competence. Note however that the individuals most involved in the Security Requirements Initiative had pre-existing technical security competence, something that may have limited the potential effect of competence building on security.

4.5 Process match

Table 5 explains the influences identified related to the match with the process of the Product Owners, the developers, and the overall culture of DevCo. The Security Requirements Initiative was viewed as an addition to existing processes, largely related to documentation, and as lacking integration with the processes of the Product Owners.

Table 5. Overview of the role of influences related to process match

Process	How this influenced the priority given to security	Influence in conceptual model (Figure 4)
Product Owners' process	• Security was viewed and represented as an add-on having separate documents in the tender, and separate Confluence pages and procedures. • Suggestions for tighter integration included having security as a requirement in Jira at the same level as other requirements (e.g., as part of Definition of Done), and including security into the Product Owners' procedures to ensure security activities were done for all projects. • Following up of the security analysis was not part of the process and there were thus no procedures in place to ensure that the analysis led to improved security in practice. • Security activities were seen as “developers' job”.	- Security viewed and represented as an add-on - Security not explicitly part of Product Owners' procedures - Follow up of security analysis is not part of the process
Development process	• Security requirements and concerns that were easy to solve and were easy to implement by developers without asking for permission to spend extra resources and time, were generally addressed. • Security was to some extent viewed as documentation and talked about as *"spending a day to document the security around that function"*. This documentation was difficult to integrate with an agile way of working and was considered to *"use up lots of time".* • Teams already found themselves drowning in detail in Jira and security documentation just added to this already existing overload of information.	+ Easy to implement by developers without asking for permission - Security viewed as documentation - Security documentation was overwhelming
Culture	• Some interviewees pointed to the role of habit, as they were generally not used to working with security this systematically. • Interviews and observations called for closer collaboration between development and operations and suggested that this could bring benefits for security due to the security competence and awareness of operations staff.	- Habit/way of working does not include security - Lack of culture for involving operations

The Security Officer experienced several challenges when it came to integrating the Security Requirements Initiative into the processes of DevCo. As seen in Figure 5, in the early stages it was important for the Security Officer to get overview of the security requirements to prepare for getting Product Owners involved in prioritisation. There was however a trade-off between presenting good quality security requirements documentation and getting early involvement from Product Owners on security. Throughout, it was challenging to find a way to structure the security documentation so that it was more usable for the Product Owners, and still provided overview for the Security Officer.

To improve the Security Requirements Initiative, interviewees suggested both to have more formal procedures for security and to have everybody take on more responsibility for security without having to add a lot of extra overhead. To explore these potential axis of integration and responsibility, Figure 8 uses Situational Analysis and its Positional Map (Clarke et al. 2016) to span out the practices that were applied in the project or that were proposed by interviewees. The adopted approach to have a separate Confluence page for security requirements represented a less integrated approach than what was suggested by some interviewees, namely, to have security more integrated into Jira and have it handled as any other requirement. However, having a separate Confluence page for security eased overview and follow up by security experts external to the project, thus supporting an approach with more central control in form of procedural requirements and security audits. The Security Requirements Initiative however struggled with limited formal authority in procedures and thus, in practice, became quite reliant on individual initiative in a context where security was largely seen as an addition.

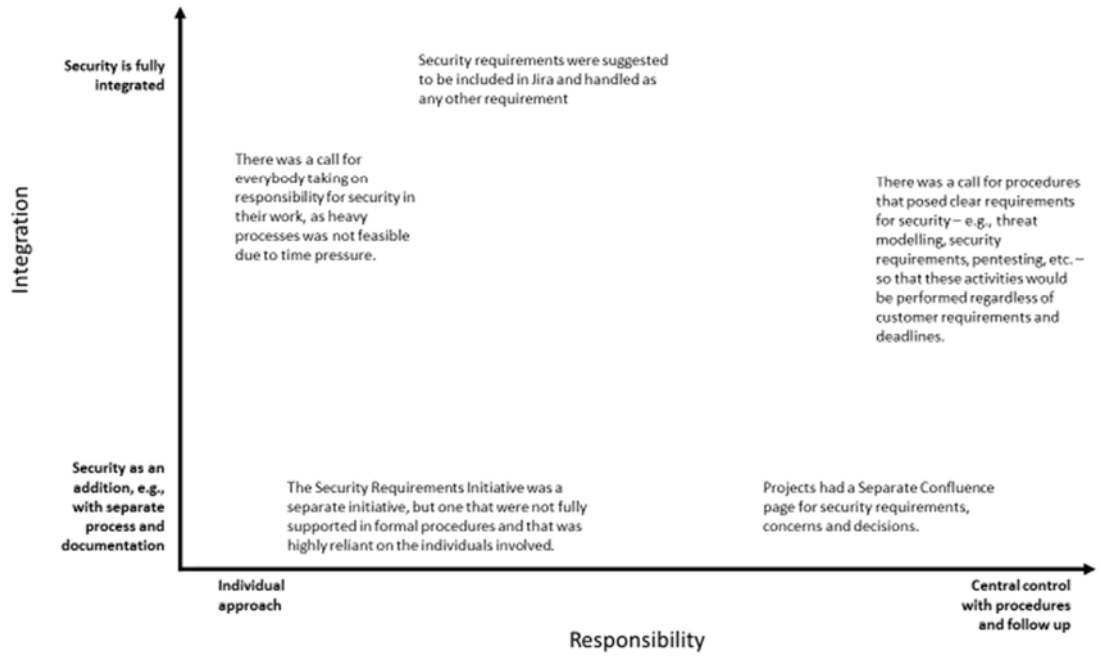

Figure 8. Positional Map showing how the practices of ProjectAlpha and the suggestions for improvements brought up in interviews can be placed according to the axis of integration and responsibility.

4.6 Influence interactions

Note that it was the combination of the influences that resulted in the changing security priorities in the projects. In each situation, several influences were present simultaneously and together contributed to or hindered the prioritisation of security. In the following we provide two examples of narratives from interviews (shortened and paraphrased) that illustrate how different influences could play together and lead to a certain security prioritisation. Both narratives come from ProjectAlpha.

The first example concerns how the development team and its Security Champion responded to a customer requirement on security:

We found that one of the things the customer had suggested was not easy to do, and it did not help much with security. In previous projects, I had wanted to implement an encryption solution and, because of this requirement, considered this project an opportunity to go through with it. I brought the issue up with the other developers using Slack, and the others agreed to this solution. Then we

just did it without involving anyone else. What we implemented is something that gives good security. However, it is probably only us developers that know it has been done as it is not documented anywhere.

This example shows the importance of developers and the Security Champion as *driving forces* for security, and the importance of their *motivation* (want to implement encryption) and *room to manoeuvre* (skills and time) that allows them to identify and implement security solutions. In this case, the *process match* was related to their ability to do this security improvement within their current process without involving other decision makers. The security requirement from the customer sparked the *visibility* of the issue and contributed to the opening (*motivation*) to go through with it, despite the customer requirement not being considered a good one. The resulting challenge related to *visibility* is not a challenge for security prioritisation regarding this particular security solution but may represent a missed opportunity to increase visibility more generally and thus influence security prioritisation more broadly.

The second example concerns a security weakness identified by the development team and the technical Product Owner and their decision on how to address this weakness:

We identified a potential security weakness. This weakness is not something we consider a major issue but is still something that could open up for some potential attacks. Several suggestions for handling this issue came up and were discussed. We settled on accepting the weakness but identified ways to address this in operations. I am however not sure if we ended up sharing these considerations and the suggestion for a solution with operations.

This example also shows the importance of developers as *driving force* for security, including their *motivation* and skills (*room to manoeuvre*). The team was capable of identifying the issue, assessing its importance, evaluating alternative solutions, and reaching a decision. However, limited *visibility* of the decision and organisational silos (*process match*) likely hindered the decision to be followed up on in practice.

5 Related work

We are not aware of other studies that aim to understand influences on security priority through studying industrial projects in depth. Thus, our study represents new knowledge on what supports and hinders prioritisation of security in practice. There are however some related studies that identify challenges, triggers and barriers, or coping strategies related to security requirements and security prioritisation more broadly, disconnected from a particular project. Terpstra et al. (2017) studied practitioners' postings on LinkedIn to understand what contextual factors practitioners perceive as challenging when it comes to security requirements in ASD. They identified 21 concepts that indicated problems and 15 coping strategies. Tøndel et al. (2017) studied software security practices of public companies to understand how to take a risk-centric approach to software security. They identified triggers and barriers for software security activity. Daneva and Wang (2018) studied documented security requirements engineering frameworks for ASD, that were known to be used in practice, and identified 46 coping strategies. As can be seen from Table 6, these match well with the influence categories we identified in our study, indicating that the conceptual model of influences on the security priority that came out of our study is relevant more broadly. Further it builds confidence in the findings in this study. This confidence is further strengthened as these influence categories are widely present also in the broader secure software engineering literature.

Table 6. Mapping of influence categories to influences identified in related studies (for Terpstra et al. and Daneva and Wang, the numbers in the table are the same IDs that are used in their papers respectively; a (-) represents a problem or barrier, a (+) represents a trigger or coping strategy).

Influence category	Terpstra et al. (2017)	Tøndel et al. (2017)	Daneva and Wang (2018)
Driving force	(-) The product owner can be a hindrance (C13, 15)	(-) Unclear responsibilities. (-) Architects do not take on responsibility.	(+) Add security champion, security master, etc. (S19, 22)
Visibility	(-) Security requirements that are poorly defined (C8). (-) Security is forgotten (C7, 18). (-) Rely on tacit knowledge (C17). (+) Use cross-functional streams to not forget (S13) (+) Check, review (S14-15)	(+) Security included as user stories	(+) Document security requirements, security debt, decisions, etc. (S1-3, 6-8, 13, 16, 38, 41) (+) Monitor, test, review, certify (S31, 33, 37, 39)
Motivation	(-) Unclear business value (C1, 3). (-) Low customer priority (C6). (-) Fight not worth fighting (C4). (-) Developers do not care (C11). (+) Use regulation to justify requirements (S7). (+) Ensure product owner support (S12).	(-) Security viewed as primarily a technical issue. (-) Limited interest from architects. (-/+) Risk perception. (+) Legal requirements. (+) Errors made. (+) Security made the project interesting.	(+) Security stakeholder in the team translates security requirements into business value (S20). (+) Own security requirements (S32). (+) Discuss the risk to the business (S43).
Room to manoeuvre	(-) Cost because of expert involvement (C2). (-) Limited knowledge among developers (C11-12, 20) and product owners (C14). (+) Add security expert to team (S9) (+) Educate and raise awareness (S10-11)	(-) Time pressure. (-) Limited competence and awareness (procurers, developers), and limited training. (+) Budget for security.	(+) Allocate time (S4) (+) Add security roles to the team (S21, 23) (+) Use risk analysis to build awareness (S26) (+) Virtual security group (S27) (+) Training (S28-30) (+) Approve/ban tools/functions (S35-36)
Process match	(-) Late security requirements (C9). (-) Planning sessions without quality stakeholders (C10).	(-) Contractors responsible – limited follow up. (-) Agile development. (-) CISO not able to follow up	(+) Integrate security requirements (S1-3, 6-8, 17) (+) Integrate security activities (S9-10, 12, 15, 40, 42)

	(-) Organisational structure (C21). (-) Limitations of agile (C19) and its implementation (C16). (+) Integrate security requirements (S1, 3-6) (+) Include security in estimates (S2)	(+) New product.	(+) Have periodic security sprints (S5) or a sprint security bucket (S11, 18)) (+) Have security experts take part in prioritisation (S24-25) (+) Let security specialists use the same whiteboards/ as the team (S34). (+) Security control posts in the process (S44-45). (+) Hybrid security and functionality testing (S46)
Other	(-) Different people prioritise security differently (C5).	(-) Balancing security with other needs	

Driving force is related to championing, and in literature the role of Security Champion (Antukh 2017; Jaatun and Cruzes 2021) utilises that terminology. In this case study, we consider championing in the context of a broader set of roles (Kocksch et al. 2018) and include championing that is not formally recognised and made visible (e.g., developer initiatives for security). As a cross-cutting concern, security requires "intertwined and distributed responsibilities, often crossing organizational, professional or even legal boundaries" (Kocksch et al. 2018). At the same time, unclear responsibilities (Kocksch et al. 2018) can be a hindrance for effective security work. Literature has shown a great potential for security experts to increase developers' sense of responsibility for security through their interaction (Xiao et al. 2014; Palombo et al. 2020) but has also pointed to challenges in this relation (Ashenden and Lawrence 2016; Tøndel et al. 2020c; Weir et al. 2020b) and the possibility of having the opposite effect if developers feel judged or security becomes a hurdle (Ashenden and Lawrence 2016). Product Owners have been identified as a potential hindrance for software security being prioritised (Terpstra et al. 2017; Alsaqaf et al. 2019).

Regarding **visibility**, literature points to the potential invisibility of security in technology and in the development work. Security has some inherent aspects that can make it invisible in development. For instance, security vulnerabilities rarely affect normal use, and thus “tend to remain invisible until one specifically looks for them” (Türpe 2017). Security can be considered a type of care work, and care is generally "not a task in itself" (Kocksch et al. 2018). The common oscillations between security and insecurity (Kocksch et al. 2018) gives security a transitory and changing appearance, and the bigger concept of software quality is considered "fuzzy" in software development (Karhapää et al. 2021). In periods of time pressure, visual cues in the workplace can be one of several strategies to increase the likelihood that security is remembered (Chowdhury et al. 2020). Further, previous studies found that it is important to have security as an explicit requirement, rather than implicit (Bartsch 2011; Poller et al. 2017; Terpstra et al. 2017).

When it comes to **motivation,** security requires ongoing commitment, as one is dealing with "a never-ending cycle of leak and fix" where security weaknesses may spark security work and where efforts to increase security can lead to new insecurities (Kocksch et al. 2018). This commitment cannot be confined to specific roles. According to Viega (2020), software vendors need outside pressure to do a good job on security. This is supported by by Xie et al. (2011), identifying customer concerns, government regulations, and organisational policies as factors that motivate or constrain

security. Security can be hard to sell as a business value and people can "drop security because they perceive it a fight not worth fighting" (Terpstra et al. 2017). The most cited reasons for not paying off technical debt have been found to be low priority, lack of organisational interest, focusing on short term goals, and cost (Freire et al. 2020). A view of security as an addition (not part of working software) and extra cost (Heijden et al. 2018) or a view of security as a hygiene-factor (where meeting security requirements will not result in positive feedback) (Loser and Degeling 2014) can reduce motivation for security.

As for **room to manoeuvre**, literature supports the importance of time, budget, and competence. Time pressure is a well-documented challenge for security work (Bartsch 2011; Poller et al. 2017; Alsaqaf et al. 2019; Behutiye et al. 2020; Chowdhury et al. 2020). Software security work can represent substantial effort (Venson et al. 2019). A recent study of human cybersecurity behaviour (Chowdhury et al. 2020) confirmed the influence of time pressure, and identified six patterns of non-secure behaviour stemming from time pressure: avoiding, bypassing, disclosing, disregarding, influencing, and over-relying. Further, security is a specialized competence involving complex reasoning (Türpe 2017). It involves the idea of an attacker, something that requires developers to think "outside the box" (Weir et al. 2020b). It is unrealistic to expect the average developer or Product Owner to be a security expert (Viega 2020). Training has been identified as a pilar for security requirements integration in ASD (Türpe and Poller 2017; Daneva and Wang 2018).

Regarding **process match,** several works point towards the need for alignment with the development processes (Türpe and Poller 2017; Daneva and Wang 2018; Tøndel and Jaatun 2020), exemplified by the following quote: "in order to succeed, approaches to encourage and anchor security work in a development setting must be aligned to the setting's defining organizational aspects" (Türpe and Poller 2017). Companies need to balance agility with the need to produce extra artefacts, perform additional activities, and have additional roles (Daneva and Wang 2018). Previous literature points to the need for setting up the company to properly react to and handle quality requirements (Olsson et al. 2019), and identifies challenges of integrating security and other quality aspects into ASD (Oueslati et al. 2015; Alsaqaf et al. 2017; Behutiye et al. 2020).

6 Discussion

The conceptual model of influences on security priority represents a way to approach how an ASD project and its context supports software security getting prioritised. For security professionals, the influence categories can be used to assess the situation in and around a project to consider the need for follow-up and select a strategy that matches the situation of a project. In the following we build on the influences identified in Section 4 and highlight the main lessons learned for each influence category. These have been identified by considering Table 1 to Table 5 as well as the experiences made by the Security Officer, extracting the key takeaways. Figure 9 gives an overview of our recommendations for what security experts should take into consideration for each of the identified influence categories.

The suggestion for process match, that companies consider whether to go towards a distributed or centralised approach to software security, is related to Figure 10. This figure is a more general form of the Positional Map already presented in Figure 8, and we use it to provide our understanding of what certain positions in the map would entail. In a *distributed* approach, security is fully integrated with the way of working and everybody takes responsibility for security (upper-left). In a *centralised* approach, one relies on procedures and separate follow-up on security (lower-right). It is unclear

what would be the added benefit of combining procedures with fully integrated security (upper-right) although this seems to be called for to some extent in the data material. Both distributed and centralised approaches to security have their merits, and can be combined, e.g., as captured by the term ambidextrous security (Cruzes and Johansen 2021). However, although software security approaches can find themselves on a continuum between being centralised and distributed, these approaches are to some extent in conflict – as experienced in our study and as also pointed out by Jarzębowicz and Weichbroth (2021). They state that some of the documentation practices related to non-functional requirements in agile software development are mutually contradictory; you cannot both have separate documentation techniques for non-functional requirements and document them in the same way and together with functional requirements.

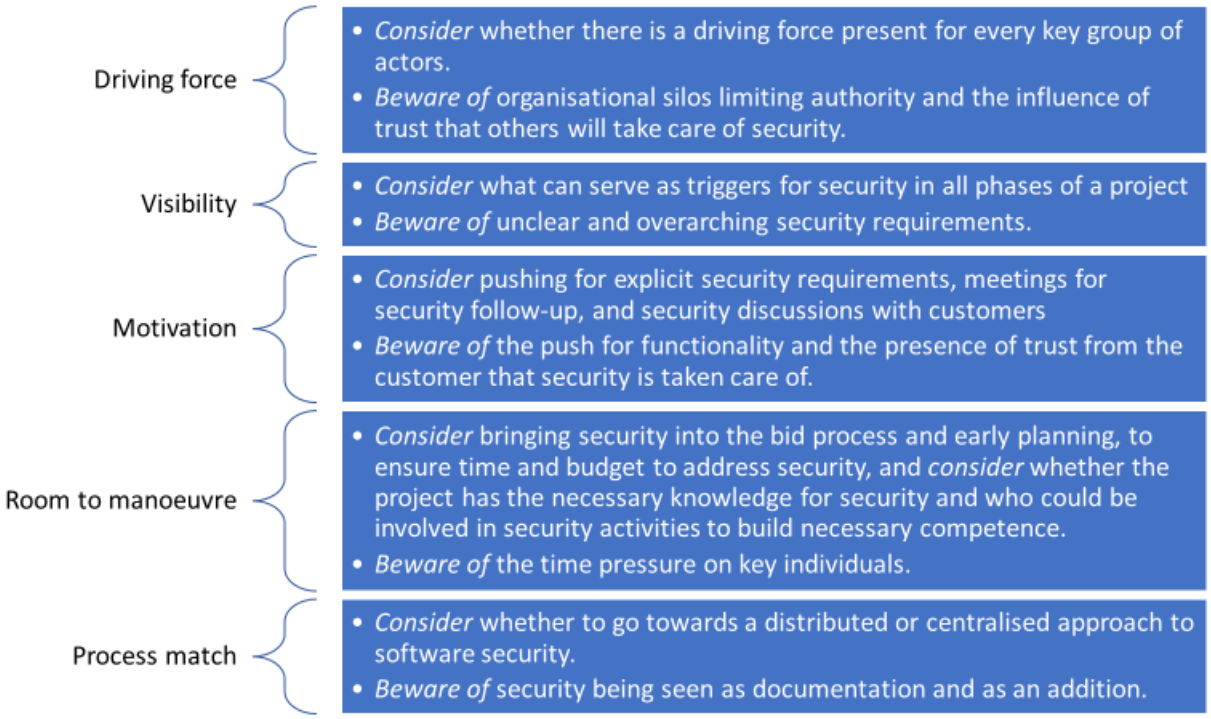

Figure 9. Recommendations for security experts for each of the influence categories, based on experiences from this case

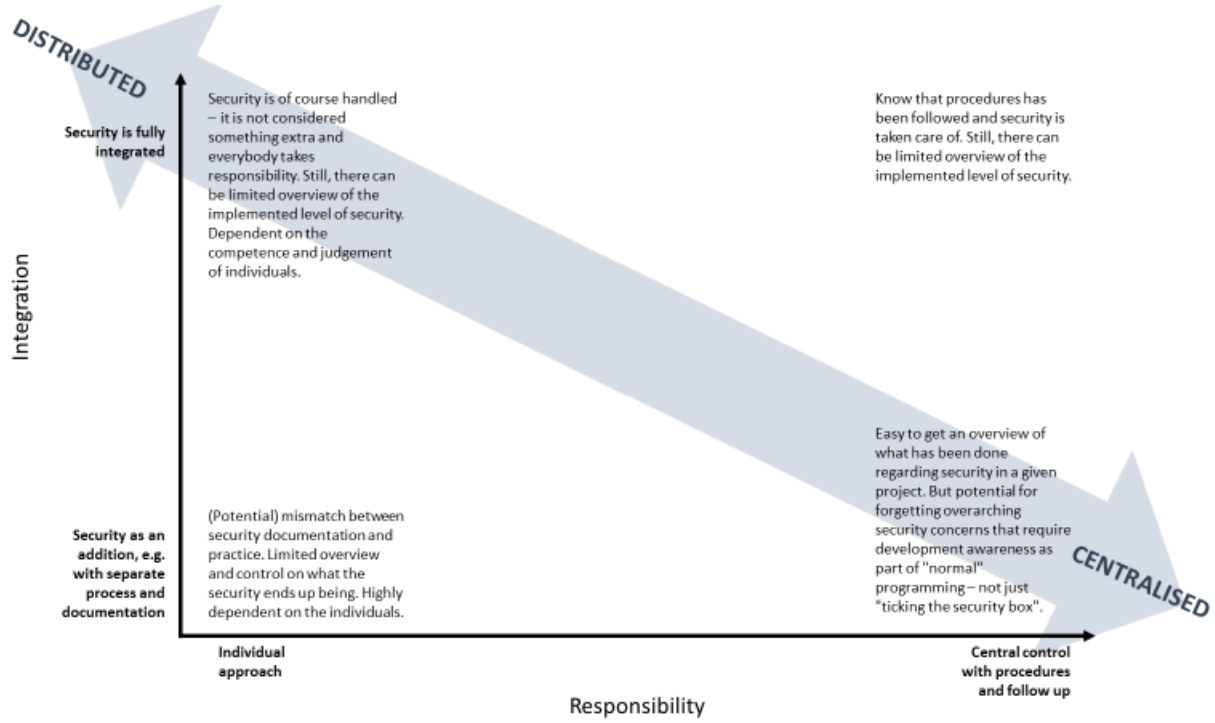

Figure 10. Positional Map showing positions that can be taken related to placement of responsibility for security and level of integration of security into development. The emphasis is on potential consequences of the various positions for security.

Note that in this study we have not investigated the influence of ASD on the priority given to security, although we place our study in the context of ASD. While literature documents challenges to software security in ASD, priority of quality aspects such as security is challenging also outside of ASD (Blaine and Cleland-Huang 2008). Security professionals need to deal with the challenges of getting security prioritised regardless of whether the challenges are caused by ASD or are more general. Still, we speculate, based on Figure 10, that a decentralised approach is particularly relevant with ASD. For SMEs with limited central security expertise, it may also be the only viable option.

We view the influence model presented in this article as an important complement to existing maturity models for software security, such as BSIMM and OWASP SAMM, that are more activity oriented, and favours documented processes. We do not oppose using BSIMM or OWASP SAMM for agile development – on the contrary we have used BSIMM in previous work (Jaatun et al. 2015; Jaatun 2017). Both BSIMM and OWASP SAMM give an overview of activities that should be considered by software development projects. However, based on our use of BSIMM with SMEs doing software development, it is our impression that the full set of activities offered can be quite overwhelming and that it is hard for an SME with limited central resources dedicated to security to get to a point where such security activities are integrated into the work processes. Further, agile principles point to "Individuals and interactions over processes and tools", "Working software over comprehensive documentation", "Customer collaboration over contract negotiation", and "Responding to change over following a plan" (Beck et al. 2001). Although this clearly does not mean that ASD calls for no processes or no documentation, it still points to a need to consider other options. Self-managed and autonomous teams are important in ASD, and as was illustrated in Figure 10, it is possible to opt for a more distributed approach to software security where more responsibility is given to the teams. However, this calls for other ways to assess projects for security and support them in their security work, and this work represents one step in that direction.

The influence model identified in this study come from the study of a case that would be placed in the more undesirable lower left quadrant of Figure 10. One should expect that the influences could have been different if the studied case had fitted elsewhere in this Positional Map. Thus, despite broad support in literature for the identified influence categories, more research is needed to get to a model of influences that we can consider to be generally applicable. We speculate that having a model of influences for security priority in ASD projects is more important for organisations aiming for a distributed approach to software security, and thus studies should be performed for such organisations.

7 Threats to validity

In the following we discuss our study in relation to the validity criteria that have been recommended for case studies within software engineering (Runeson and Höst 2009; Yin 2018).

7.1 Construct validity

Construct validity can be defined as the "accuracy with which a case study's measures reflect the concepts being studied" (Yin 2018). This study is concerned with the concept of security priority. Through identifying what influences the priority given to security in an ASD project, it contributes to operationalising what security priority can mean in this context.

According to Sjøberg and Bergersen (2021), threats to construct validity can be divided into three categories: inadequate definition of the concept, construct underrepresentation, and construct-representation bias. Literature did not offer us a theory of software security priority to build on in our study. However, literature offered descriptions of security work as cross-cutting and fuzzy (Türpe 2017; Kocksch et al. 2018; Karhapää et al. 2021), and identified a broad set of challenges to security in ASD (Inayat et al. 2015; Oueslati et al. 2015; Khaim et al. 2016; Alsaqaf et al. 2017; Behutiye et al. 2020; Jarzębowicz and Weichbroth 2021). Thus, in the design of the case study we were deliberately broad and open in our view of what security priority could look like and what could influence the security priority. We aimed to enter the interviews and observations and our study of documentation with an open mind. In observations we noted down as much detail as possible – to reduce risk that we missed important aspects whose significance we did not understand at the time of observation. The use of Situational Analysis (Clarke et al. 2016) supported us in taking in the totality of the case and all its elements. Thus, we did address the threat of construct underrepresentation and construct-representation bias in our design, while there remains a risk of inadequate definition of the concept; 'security priority' was difficult to define and thus difficult to measure.

Additional strategies important to support construct validity was our prolonged involvement with DevCo, the use of triangulation in data collection and analysis, the involvement of the Security Officer of DevCo in discussing the findings, and the opportunity given to DevCo representatives to review the initial research report. This way, we ensured that the concept of security priority was addressed from different angles and over time, and that our understandings as researchers reflected the understanding and experiences of company representatives.

7.2 Internal validity

Internal validity considers the "strength of the causal or other "how" and "why" inferences made in a case study" (Yin 2018). Strategies supporting internal validity in qualitative studies include thick and context-rich descriptions, triangulation, linking results to prior or emerging theory, identify areas of uncertainty, seeking negative evidence, considering rival explanations, and original participants

finding the conclusions accurate (Miles et al. 2018). In this study we have made use of all these strategies. Further, our analysis approach supported us in identifying similarities as well as discrepancies in the data, as we in turn focused our analysis efforts on different parts of the data material. Note that though we made efforts to look for discrepant evidence and question our findings, we did not find much. In the following we point to the main potential biases we have identified.

We collected the perspectives of individuals in different roles and with different levels of competence on security. There are however several additional actors whose voices we did not seek out in data collection but that could have given valuable insights and perspectives. Examples are developers, managers, testers, and operations.

Our close collaboration with the Security Officer, both during data collection and analysis, was a strength and a weakness in our study. As we gained access to the case through the Security Officer and collaborated closely with the Security Officer throughout, there was a threat of us favouring the Security Officer perspective and being seen as too strongly linked with the Security Officer. We were concerned that this could lead to a potential unwillingness to share information with us in interviews. Thus, in interviews we brought up these issues in the beginning, as part of going through a data consent form, and made the interviewees aware that information they shared in the interview could be withheld from the Security Officer. However, none of the interviewees expressed any concerns related to us sharing information with the Security Officer, and in our impression, talked quite freely also about their challenges. This made us conclude that our strong relation with the Security Officer was viewed positively, increasing our potential contribution to the work in the company, and increasing their trust in us as researchers.

It is likely that our presence as researchers influenced the project we studied. At the very least, it is likely that our presence worked as a visual cue to consider and talk about security issues. To make us aware of and reflect on our influence as researchers on the case, we specifically included this as a point in the observation template. Thus, for every observation the first author wrote about how she may have influenced what she observed, and this regular reflection enabled us to take this aspect into account in data collection and analysis. In interviews, our influence as researchers was probably even larger, as we together with the interviewees steered the conversation and were co-creators in the narratives that came out of the interviews. However, we were aware of our influence and took measures, in the creation of the interview guide and during the interviews themselves, to create an atmosphere of trust and allow the interviewee to talk freely about their experience related to specific examples of their choosing.

7.3 External validity

External validity concerns the "extent to which the findings from a case study can be analytically generalized to other situations that were not part of the original study" (Yin 2018). Some particulars of the studied case may have led this case to experience different influences than what would be present in most other development projects. Regarding *Driving force,* it is likely that different influences would have been observed if there had not been a Security Champion in the team, and if the Security Officer had not been placed in the development department. When it comes to *Visibility*, different influences might have been experienced if, e.g., security had been part of procedures at the Product Owner level. Regarding *Motivation*, the security push from the customer and the third-party security experts involved on the customer side in ProjectAlpha will not be present in all development projects – and was not present in ProjectBeta. For *Room to manoeuvre*, the need to compete for projects in a bid process contributed to challenges regarding time and budget for security. This may be different for other types of projects. Further, interviews revealed that the Product Owners were

particularly pressed for time during the period of the study because of several big projects in parallel. And finally, for *Process match*, there was a lack of culture for involving operations and security was not fully part of their processes. They were also already being overloaded with documentation in Jira. This may not be the case for other projects, though we hypothesise that many projects could experience similar challenges.

To support external validity, we have aimed for a thick description to support readers in judging whether the reported results are relevant for other contexts. Furthermore, we show how our results confirm the results from previous studies (see Section 5). This need for a thick description however had to be balanced with the wish for anonymity from the studied company. Norway is a small country, thus too many details could easily jeopardize this anonymity.

7.4 Reliability

Reliability can be defined as the "consistency and repeatability of producing a case study's findings" (Yin 2018). Throughout we kept an overview of all data collection activities and interaction with the case. In data collection, we emphasised the collection of detailed data, recording and transcribing all the interviews and making thorough observation notes. We kept an analysis journal and used several techniques in analysis to ensure a thorough consideration of all the evidence. Findings were validated with the Security Officer of DevCo.

8 Conclusion

In this article we have reported on the findings from a longitudinal case study that gives insight into the strategy used by one security professional in an SME to influence the priority assigned to security in software development projects in the company. Based on the study of this case we have identified influences on the priority given to security in the development projects, as well as made recommendations for security professionals who want to increase key decision makers' attention to security. With this work we complement existing maturity models that are activity oriented with a model of influence categories giving an overview of characteristics of situations that can influence the priority given to security.

For practitioners, this study offers knowledge of potential influences on the priority given to software security in ASD. This knowledge can be used to assess which ongoing development projects need to be given particular attention to increase the chances that software security is adequately addressed throughout. Further, this assessment considers that projects may do well on security despite a lack of formal procedures or processes. The recommendations can support security professionals in selecting successful strategies for influencing ASD projects and the priority they give to security. As part of selecting such a strategy, we suggest that companies should make a strategic decision whether they, on a longer-term, would aim to evolve towards a centralised or a distributed approach to security. When applying the results from this article, practitioners should however be aware that the results are based on a single case study. Thus, it is important to consider how well the studied project matches with the context where this result is applied.

For researchers, this study provides a model of influences on the priority given to software security, based on a detailed study of one case over a longer period, and is built on an analysis approach that takes the full situation into account. This model needs to be validated and extended upon in further research. The influences identified, as well as the recommendations for security practitioners, can

provide a basis to build upon to in research aimed to make recommendations for security professionals on which strategy to choose in varying circumstances. Moreover, researchers could take the discussions we provide on centralised vs. distributed security approaches as inspiration to conduct studies aimed towards making recommendations for when companies should select a distributed vs. a centralised approach to security. Similarly, future studies could be aimed towards providing companies with recommendations on how to gradually build successful distributed and centralised security initiatives in various contexts, as well as when and how to combine features of distributed and centralised approaches successfully.

Acknowledgements

This research was funded by the Research Council of Norway, grant nr. 247678. The authors would like to thank the company where this study was performed, especially the interviewees and the participants in the meetings we observed. We would also like to thank our colleagues in the "Fabrikk" for valuable comments on an earlier version of this article.

References

Alsaqaf W, Daneva M, Wieringa R (2017) Quality requirements in large-scale distributed agile projects–a systematic literature review. In: Grünbacher P. , Perini A. (ed) Requirements Engineering: Foundation for Software Quality (REFSQ 2017), Springer, Cham, vol 10153, pp 219-234. https://doi.org/10.1007/978-3-319-54045-0_17

Alsaqaf W, Daneva M, Wieringa R (2019) Quality requirements challenges in the context of large-scale distributed agile: An empirical study. Information and software technology 110:39-55. https://doi.org/10.1016/j.infsof.2019.01.009

Antukh A (2017) Security champions playbook. OWASP. https://github.com/c0rdis/security-champions-playbook/tree/master/Security%20Playbook

Ashenden D, Lawrence D (2016) Security dialogues: Building better relationships between security and business. IEEE Security & Privacy 14.3:82-87. https://doi.org/10.1109/MSP.2016.57

Bakalova Z, Daneva M, Herrmann A, Wieringa R (2011) Agile requirements prioritization: What happens in practice and what is described in literature. International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer. https://doi.org/10.1007/978-3-642-19858-8_18

Baldassarre MT, Barletta VS, Caivano D, Piccinno A (2021) Integrating Security and Privacy in HCD-Scrum. CHItaly 2021: 14th Biannual Conference of the Italian SIGCHI Chapter. Bolzano, Italy. pp Article 37. https://doi.org/10.1145/3464385.3464746

Bartsch S (2011) Practitioners' perspectives on security in agile development. 2011 Sixth International Conference on Availability, Reliability and Security (ARES). pp 479-484. https://doi.org/10.1109/ARES.2011.82

Beck K, Beedle M, Van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R. (2001) Manifesto for agile software development. https://agilemanifesto.org/

Behutiye W, Karhapää P, López L, Burgués X, Martínez-Fernández S, Vollmer AM, Rodríguez P, Franch X, Oivo M (2020) Management of quality requirements in agile and rapid software development: A systematic mapping study. Information and software technology 123:106225. https://doi.org/10.1016/j.infsof.2019.106225

Blaine JD, Cleland-Huang J (2008) Software Quality Requirements: How to Balance Competing Priorities. IEEE Software 25.2:22-24. https://doi.org/10.1109/MS.2008.46

Chowdhury NH, Adam MTP, Teubner T (2020) Time pressure in human cybersecurity behavior: Theoretical framework and countermeasures. Computers & Security 97:101931. https://doi.org/10.1016/j.cose.2020.101931

Clarke AE, Friese C, Washburn R (2016) Situational analysis in practice: Mapping research with grounded theory. Routledge.

Cruzes DS, Johansen EA (2021) Building an Ambidextrous Software Security Initiative. In: Manuel M., Jorge Marx G., Rory V. O. C. , Alena B. (ed) Balancing Agile and Disciplined Engineering and Management Approaches for IT Services and Software Products, IGI Global, pp 167-188. https://doi.org/10.4018/978-1-7998-4165-4.ch009
Daneva M, Wang C (2018) Security Requirements Engineering in the Agile Era: How Does it Work in Practice? 2018 IEEE 1st International Workshop on Quality Requirements in Agile Projects (QuaRAP). https://doi.org/10.1109/QuaRAP.2018.00008
Eisenhardt KM (1989) Building theories from case study research. Academy of management review 14.4:532-550. https://doi.org/10.5465/amr.1989.4308385
Freire S, Rios N, Gutierrez B, Torres D, Mendonça M, Izurieta C, Seaman C, Spínola RO (2020) Surveying software practitioners on technical debt payment practices and reasons for not paying off debt items. The Evaluation and Assessment in Software Engineering (EASE). pp 210-219. https://doi.org/10.1145/3383219.3383241
Heijden Avd, Broasca C, Serebrenik A (2018) An empirical perspective on security challenges in large-scale agile software development. Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. Oulu, Finland. https://doi.org/10.1145/3239235.3267426
Inayat I, Salim SS, Marczak S, Daneva M, Shamshirband S (2015) A systematic literature review on agile requirements engineering practices and challenges. Computers in human behavior 51:915-929. https://doi.org/10.1016/j.chb.2014.10.046
Ionita D, Velden Cvd, Ikkink H-JK, Neven E, Daneva M, Kuipers M (2019) Towards risk-driven security requirements management in agile software development. International Conference on Advanced Information Systems Engineering, Springer. https://doi.org/10.1007/978-3-030-21297-1_12
Jaatun MG (2017) The Building Security in Maturity Model as a Research Tool. Empirical Research for Software Security: Foundations and Experience.
Jaatun MG, Cruzes DS (2021) Care and Feeding of Your Security Champion. 2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), IEEE. https://doi.org/10.1109/CyberSA52016.2021.9478254
Jaatun MG, Cruzes DS, Bernsmed K, Tøndel IA, Røstad L (2015) Software Security Maturity in Public Organisations. In: Lopez J. , Mitchell C. (ed) Information Security, ISC 2015, Springer, Cham, vol 9290, pp 120-138. https://doi.org/10.1007/978-3-319-23318-5_7
Jarzębowicz A, Weichbroth P (2021) A Systematic Literature Review on Implementing Non-functional Requirements in Agile Software Development: Issues and Facilitating Practices. In: Przybyłek A., Miler J., Poth A. , Riel A. (ed) Lean and Agile Software Development, LASD 2021, Springer, Cham, vol 408, pp 91-110. https://doi.org/10.1007/978-3-030-67084-9_6
Karhapää P, Behutiye W, Rodríguez P, Oivo M, Costal D, Franch X, Aaramaa S, Choraś M, Partanen J, Abherve A (2021) Strategies to manage quality requirements in agile software development: a multiple case study. Empirical Software Engineering 26.28. https://doi.org/10.1007/s10664-020-09903-x
Khaim R, Naz S, Abbas F, Iqbal N, Hamayun M (2016) A Review of Security Integration Technique in Agile Software Development. International Journal of Software Engineering & Applications (IJSEA) 7.3:49-68.
Koç G, Aydos M (2017) Trustworthy scrum: Development of secure software with scrum. 2017 International Conference on Computer Science and Engineering (UBMK), IEEE.
Kocksch L, Korn M, Poller A, Wagenknecht S (2018) Caring for IT Security: Accountabilities, Moralities, and Oscillations in IT Security Practices. Proc. ACM Hum.-Comput. Interact. 2.CSCW. https://doi.org/10.1145/3274361
Loser K-U, Degeling M (2014) Security and privacy as hygiene factors of developer behavior in small and agile teams. In: Kimppa K., Whitehouse D., Kuusela T. , Phahlamohlaka J. (ed) ICT and Society, HCC 2014, Springer, Berlin, Heidelberg, vol 431, pp 255-265. https://doi.org/10.1007/978-3-662-44208-1_21

Maxwell JA (2013) Qualitative research design: An interactive approach. Applied Social Research Methods Series, 41. Sage publications.
McGraw G (2006) Software security: building security in. Addison-Wesley Professional.
Migues S, Erlikhman E, Ewers J, Nassery K (2021) BSIMM12 2021 Foundations Report. Synopsis. https://www.bsimm.com/content/dam/bsimm/reports/bsimm12-foundations.pdf
Miles MB, Huberman AM, Saldaña J (2018) Qualitative data analysis: A methods sourcebook. Sage publications.
Olsson T, Wnuk K, Gorschek T (2019) An empirical study on decision making for quality requirements. Journal of Systems and Software 149:217-233. https://doi.org/10.1016/j.jss.2018.12.002
Olsson T, Wnuk K, Jansen S (2021) A validated model for the scoping process of quality requirements: a multi-case study. Empirical Software Engineering 26.2:1-29. https://doi.org/10.1007/s10664-020-09896-7
Oueslati H, Rahman MM, Othmane Lb (2015) Literature Review of the Challenges of Developing Secure Software Using the Agile Approach. 2015 10th International Conference on Availability, Reliability and Security, 24-27 Aug. 2015. pp 540-547. https://doi.org/10.1109/ares.2015.69
Palombo H, Tabari AZ, Lende D, Ligatti J, Ou X (2020) An ethnographic understanding of software (in) security and a co-creation model to improve secure software development. Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020).
Pelrine J (2011) On Understanding Software Agility: A Social Complexity Point Of View. Emergence: Complexity & Organization 13:26-37.
Pohl C, Hof H-J (2015) Secure scrum: Development of secure software with scrum. arXiv preprint arXiv:1507.02992.
Poller A, Kocksch L, Türpe S, Epp FA, Kinder-Kurlanda K (2017) Can Security Become a Routine?: A Study of Organizational Change in an Agile Software Development Group. Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing. Portland, Oregon, USA. pp 2489-2503. https://doi.org/10.1145/2998181.2998191
Riessman CK (2008) Narrative methods for the human sciences. Sage.
Rindell K, Hyrynsalmi S, Leppänen V (2015) Securing Scrum for VAHTI. 14th Symposium on Programming Languages and Software Tools.
Rindell K, Hyrynsalmi S, Leppänen V (2017) Busting a myth: Review of agile security engineering methods. ARES '17: Proceedings of the 12th International Conference on Availability, Reliability and Security. pp 1-10. https://doi.org/10.1145/3098954.3103170
Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering 14.2:131-164. https://doi.org/10.1007/s10664-008-9102-8
Schwaber K (2004) Agile project management with Scrum. Microsoft press.
Sjøberg D, Bergersen G (2021) Construct Validity in Software Engineering. TechRxiv. https://dx.doi.org/10.36227/techrxiv.14141027.v1
Terpstra E, Daneva M, Wang C (2017) Agile practitioners' understanding of security requirements: insights from a grounded theory analysis. 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW). pp 439-442. https://doi.org/10.1109/REW.2017.54
Thomas TW, Tabassum M, Chu B, Lipford H (2018) Security during application development: An application security expert perspective. CHI '18: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. https://doi.org/10.1145/3173574.3173836
Tuladhar A, Lende D, Ligatti J, Ou X (2021) An Analysis of the Role of Situated Learning in Starting a Security Culture in a Software Company. USENIX Symposium on Usable Privacy and Security (SOUPS) 2021.
Türpe S (2017) The trouble with security requirements. 2017 IEEE 25th International Requirements Engineering Conference (RE), IEEE. https://doi.org/10.1109/RE.2017.13
Türpe S, Poller A (2017) Managing Security Work in Scrum: Tensions and Challenges. SecSE@ ESORICS 2017:34-49.

Tøndel IA, Cruzes DS, Jaatun MG (2020a) Achieving" Good Enough" Software Security: The Role of Objectivity. EASE '20: Proceedings of the Evaluation and Assessment in Software Engineering. pp 360-365. https://doi.org/10.1145/3383219.3383267
Tøndel IA, Cruzes DS, Jaatun MG (2020b) Using Situational and Narrative Analysis for Investigating the Messiness of Software Security. ESEM '20: Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). pp 1-6. https://doi.org/10.1145/3382494.3422162
Tøndel IA, Jaatun MG (2020) Towards a Conceptual Framework for Security Requirements Work in Agile Software Development. International Journal of Systems and Software Security and Protection (IJSSSP) 11.1:33-62. https://doi.org/10.4018/IJSSSP.2020010103
Tøndel IA, Jaatun MG, Cruzes DS (2020c) IT security is from Mars, software security is from Venus. IEEE Security & Privacy 18.4:48-54. https://doi.org/10.1109/MSEC.2020.2969064
Tøndel IA, Jaatun MG, Cruzes DS, Moe NB (2017) Risk centric activities in secure software development in public organisations. International Journal of Secure Software Engineering (IJSSE) 8.4:1-30. https://doi.org/10.4018/IJSSE.2017100101
van der Stock A, Cuthbert D, Manico J, Grossman JC, Lang E (2021) OWASP Application Security Verification Standard 4.0.3, ed.
van der Veer R. (2019) SAMM Agile Guidance. https://owaspsamm.org/guidance/agile/
van Wyk KR, McGraw G (2005) Bridging the gap between software development and information security. Security & Privacy, IEEE 3.5:75-79. https://doi.org/10.1109/MSP.2005.118
Venson E, Alfayez R, Gomes MM, Figueiredo RM, Boehm B (2019) The impact of software security practices on development effort: An initial survey. 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), IEEE. https://doi.org/10.1109/ESEM.2019.8870153
Viega J (2020) 20 Years of Software Security. Computer 53.11:75-78. https://doi.org/10.1109/MC.2020.3013685
Weir C, Becker I, Noble J, Blair L, Sasse MA, Rashid A (2020a) Interventions for long-term software security: Creating a lightweight program of assurance techniques for developers. Software: Practice and Experience 50.3:275-298. https://doi.org/10.1002/spe.2774
Weir C, Rashid A, Noble J (2020b) Challenging software developers: dialectic as a foundation for security assurance techniques. Journal of Cybersecurity 6.1. https://doi.org/10.1093/cybsec/tyaa007
Williams L, Meneely A, Shipley G (2010) Protection Poker: The New Software Security "Game". IEEE Security and Privacy 8.3:14-20. https://doi.org/10.1109/msp.2010.58
Xiao S, Witschey J, Murphy-Hill E (2014) Social influences on secure development tool adoption: why security tools spread. Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing. pp 1095-1106. https://doi.org/10.1145/2531602.2531722
Xie J, Lipford HR, Chu B (2011) Why do programmers make security errors? 2011 IEEE symposium on visual languages and human-centric computing (VL/HCC), IEEE. https://doi.org/10.1109/VLHCC.2011.6070393
Yin RK (2018) Case study research and applications, 6e. Sage.