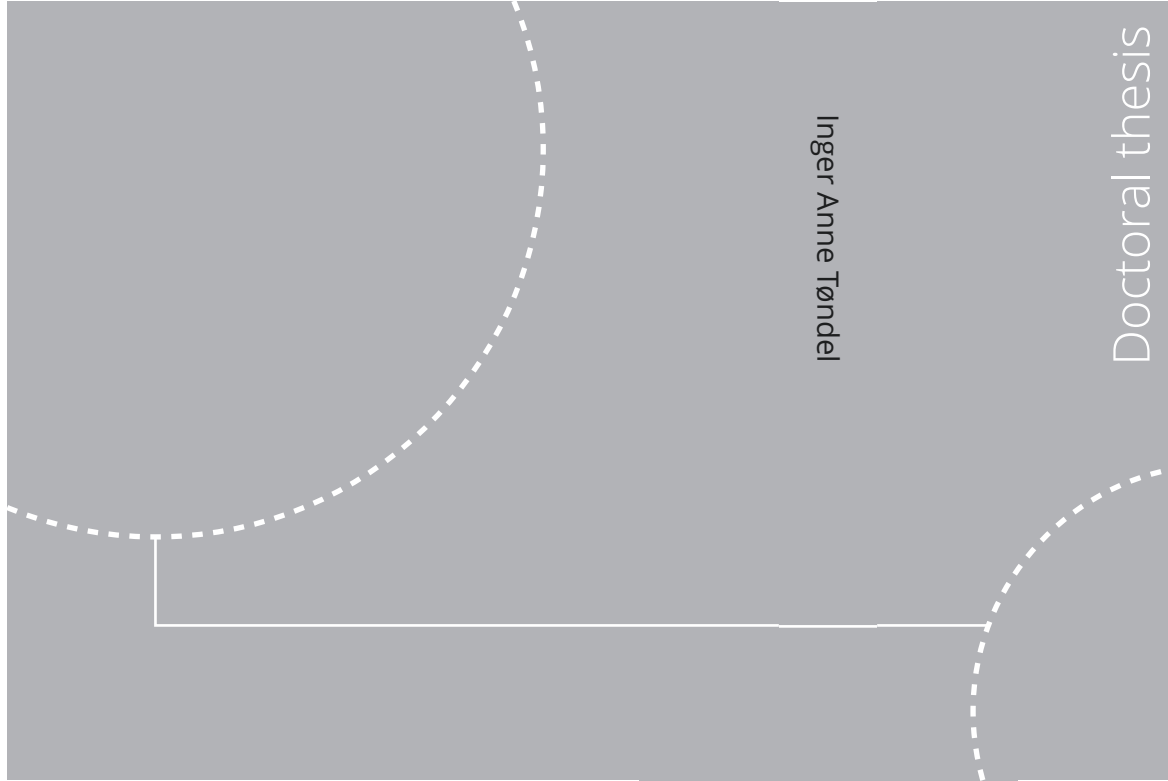


ISBN 978-82-326-6159-6 (printed ver.)  
ISBN 978-82-326-5334-8 (electronic ver.)  
ISSN 1503-8181 (printed ver.)  
ISSN 2703-8084 (electronic ver.)



Doctoral theses at NTNU, 2022:285

Inger Anne Tøndel

# Prioritisation of security in agile software development projects

Inger Anne Tøndel

# Prioritisation of security in agile software development projects

Thesis for the degree of Philosophiae Doctor

Trondheim, October 2022

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



Norwegian University of  
Science and Technology

**NTNU**

Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology  
and Electrical Engineering  
Department of Computer Science

© Inger Anne Tøndel

ISBN 978-82-326-6159-6 (printed ver.)  
ISBN 978-82-326-5334-8 (electronic ver.)  
ISSN 1503-8181 (printed ver.)  
ISSN 2703-8084 (electronic ver.)

Doctoral theses at NTNU, 2022:285



Printed by Skipnes Kommunikasjon AS

*You are a masterpiece  
fighting to be a silly selfie  
with a hideous filter*

---

Propaganda,  
*It's complicated*





*“Do not fly too high,” said Aslan.  
“Do not try to go over the tops of the  
great ice-mountains. Look out for  
the valleys, the green places, and fly  
through them. There will always be  
a way through. And now, be gone  
with my blessing.”*

---

C. S. Lewis,  
*The Chronicles of Narnia*



## ABSTRACT

Agile software development is driven by business value, and strives towards visible progress through features. Consequently, the somewhat invisible and overarching aspect of software security is at the risk of being neglected.

A key assumption of this thesis is that to achieve adequate security within acceptable costs (“good enough” security), software development projects need to be able to make priorities on what security is needed throughout development. The thesis addresses the following overall research problem: *How can regular security prioritisation be integrated into agile software development so that software products end up with a level of security that is “good enough”?* To this end, the thesis investigates 1) what influences the security prioritisation throughout an agile software development project, and 2) how security roles and activities can support an agile software development project in reaching a “good enough” prioritisation of security.

The research follows a design science approach, studying and designing process support for companies wanting to improve their software security prioritisation. The investigation is centred on small and medium sized companies developing “normal” software, i.e., software that is not security critical nor has security as a key feature of the product. The need for trade-offs and prioritisations between security and other software aspects is likely to be more pressing when security is not a main development goal, and smaller companies have been identified as having a higher potential for improvement in their software security compared to larger companies.

The thesis suggests that to improve prioritisation of security in agile software development, companies can apply regular security prioritisation meetings, and security experts in the company can be empowered with knowledge on how to influence the security priority. The foundation for this suggestion is documented in a collection of papers. The thesis offers the following main contributions that are aimed towards both practitioners and researchers: 1) A conceptual model of the influences on security priority in agile software development, 2) Identified and evaluated strategies that security experts can take in influencing the security priority of agile software development projects, 3) A new and evaluated meeting approach for continuous software security in agile software development, and 4) Rich descriptions of practical experiences with improving software security prioritisation, bridging the gap between science and practice.



## PREFACE

This thesis is submitted to the *Norwegian University of Science and Technology* (NTNU) for the partial fulfilment of the requirements for the degree of *Philosophiae Doctor* (PhD).

This doctoral work has been conducted at the Department of Computer Science (IDI), NTNU, Trondheim. In the early stages of the work, the work was done under the supervision of Professor Pekka Abrahamsson. Later, Professor Guttorm Sindre took over as main supervisor. Professor Daniela Soares Cruzes, Senior Research Scientist Martin Gilje Jaatun, Professor Jingyue Li, and Professor Colin Boyd were co-supervisors.

The work was financed by the Research Council of Norway through the project *SoS-Agile: Science of Security in Agile Software Development*, grant number 247678.



## ACKNOWLEDGEMENT

In this thesis, I have been lucky to have two skilful professors as main supervisors. Thanks to Professor Pekka Abrahamson for getting me started in a good direction, for challenging me in the definition of research questions, and for directing me towards useful and interesting courses. Thanks to Professor Guttorm Sindre for continuing the supervision and helping me finalise the work. Thanks for your positivity, for making time for meetings and discussions, and for providing useful comments on the many pages of manuscripts that I have sent your way.

I would like to offer my special thanks to my co-supervisor Professor Daniela Soares Cruzes, who encouraged me to take on this thesis work and has been a tremendous support throughout. You have helped me identify ways to improve the work and move forward in a good direction. I appreciate both your strong professional competence and your friendship. Furthermore, I would like to express my sincere gratitude to my co-supervisor, Senior Research Scientist Martin Gilje Jaatun, for collaboration and for your strong support in this thesis. During my 18 years as your colleague at SINTEF Digital, I have appreciated the many opportunities for joint work, and the way you always welcome discussions and questions, despite a busy schedule. I have learned a lot from you. Thanks also to my co-supervisors Professor Jingyue Li and Professor Colin Boyd.

Of immense importance to this work, are the many practitioners that have spent their time participating in interviews and retrospectives, and that have allowed me access to their meetings. This work would not have been possible without you, and I am honoured and deeply grateful for being allowed to study you and your companies. Thanks for your positivity and your investment in this work.

Throughout my time as a PhD candidate at IDI, NTNU, I was lucky to be part of Forskerfabrikken, where I enjoyed fruitful discussions on papers in progress and on research methods. Thanks for broadening my views, and thanks for the insightful comments and suggestions received on draft versions of several of the publications that became part of this thesis. My thanks also go to those being co-authors of the publications contributing to this thesis. I have enjoyed working with all of you, and appreciate what I have learned from our collaboration.

I would like to thank my colleagues at SINTEF for supporting me in this PhD work, offering a good working environment, and respecting my need to balance the demands within the research



projects with the need to focus on this PhD work. Thanks to Senior Research Scientist Per Håkon Meland for letting me use the latex files you used for your thesis. Thanks to Senior Research Scientist Gunnar Brataas for joint exploration of the relation between scalability and security in software development.

Of tremendous importance to this thesis is the support from family and friends. First and foremost, I would like to thank my husband, Magnus, for being there by my side, supporting me, tolerating me. I am grateful for the life we share together. Thanks also to my kids, Mia and Henrik, for enriching my life and being the awesome persons you are. Thanks to my friends and the extended family for all the support over the last years - I appreciate you and what you bring to my life.

I have been given a tremendous gift - this life, these circumstances, these opportunities. I am fully aware that I am privileged to be able to do the work I do and live the life I live. Thus, borrowing words from Gungor and their song "Crag and Clay", I thank The Giver of it all:

*"All praises to the one who made it all and finds it beautiful.*

*Fearfully and wonderfully and beautifully made."*

Inger Anne Tøndel,  
Trondheim, June 20, 2022

# CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Problem outline . . . . .	1
1.2	Research objective and design . . . . .	3
1.3	Research context . . . . .	5
1.4	Main contributions . . . . .	6
1.5	Overview of papers . . . . .	7
1.6	Structure of the thesis . . . . .	9
<b>2</b>	<b>Background</b>	<b>11</b>
2.1	Risk management, security requirements, and security prioritisation . . . . .	11
2.2	Prioritisation in agile software development . . . . .	14
2.3	Security experts and their role in adoption of software security practices . . . . .	17
<b>3</b>	<b>Research approach</b>	<b>19</b>
3.1	The design cycle . . . . .	19
3.2	The design science approach as applied in this work . . . . .	21
3.3	Use of case studies and technical action research within a design science approach . . . . .	23
<b>4</b>	<b>Contributions</b>	<b>27</b>
4.1	C1 - A conceptual model of the influences on security priority in agile software development . . . . .	27
4.2	C2 - Identified and evaluated strategies that security experts can take in influencing the security priority of agile software development projects . . . . .	30
4.3	C3 - A new and evaluated meeting approach for continuous software security in agile software development . . . . .	32
4.4	C4 - Rich descriptions of practical experiences with improving software security prioritisation, bridging the gap between science and practice . . . . .	34
<b>5</b>	<b>Discussion</b>	<b>37</b>
5.1	Research objective revisited . . . . .	37
5.2	Threats to validity . . . . .	42
5.3	Recommendations for future work . . . . .	45
<b>6</b>	<b>Conclusion</b>	<b>47</b>
	<b>References</b>	<b>49</b>
<b>A</b>	<b>Primary papers</b>	<b>59</b>

A: ‘Risk Centric Activities in Secure Software Development in Public Organisations’	61
B: ‘IT Security Is From Mars, Software Security Is From Venus’	93
C: ‘Collaborative security risk estimation in agile software development’	105
D: ‘Towards a Conceptual Framework for Security Requirements Work in Agile Software Development’	135
E: ‘The Security Intention Meeting Series as a way to increase visibility of software security decisions in agile development projects’	167
F: ‘Achieving “Good Enough” Software Security: The Role of Objectivity’	177
G: ‘Influencing the security prioritisation of an agile software development project’	185
H: ‘Continuous software security through security prioritisation meetings’	207
<b>B Secondary papers</b>	<b>239</b>

## LIST OF FIGURES

1.1	Overview of the knowledge questions . . . . .	5
2.1	Overview of the areas covered by BSIMM and OWASP SAMM . . . . .	12
3.1	The design cycle (adapted from Wieringa [14]) . . . . .	20
3.2	The design cycle iterations of this thesis. Note that for practical purposes, to distinguish the iterations in the figure, I have used two colours and the arrows representing the cycles go in different directions. This does not have any significance when it comes to the method - each arrow represents one design cycle iteration. . . . .	24
3.3	The relation between the knowledge questions, the research questions in the papers, and the design cycle iterations. Some papers have contributions to more than one knowledge question. . . . .	25
4.1	Illustration of the relation between the knowledge questions, the contributions, and the papers . . . . .	28
4.2	The progress towards contribution C1 . . . . .	29
4.3	The progress towards contribution C2 . . . . .	31
4.4	The progress towards contribution C3 . . . . .	33
5.1	Overview of the contribution using the visual abstract template for design science research [80, 81] . . . . .	38



## LIST OF TABLES

3.1	Overview of studies (CS = case study; AR = action research; LR = literature review, I = interviews) . . . . .	26
4.1	Overview of paper contributions . . . . .	36
5.1	Overview of the discussed validity threats and their relevance to the main contributions (a 'V' indicating relevance) . . . . .	42



## INTRODUCTION

In today's digitalised and interconnected world, most, if not all, software development projects need to consider security. Still, security is usually a secondary goal which can add to development time and cost. In agile software development, the emphasis on business value and visible progress through features has been found to increase the risk that security is not addressed from the beginning [1] and ends up being neglected [2–5]. However, there are limits to how much priority security should be given. Most software development projects would not need to opt for the best security possible. Furthermore, it is not always the case that the more time and money you spend on security, the better your security will be. It is quite possible to spend money and effort on the wrong things. Thus, prioritising security is not only prioritising to do security, but prioritising among all the potential things to do.

This section introduces the problem of security prioritisation within an agile software development context. It explains the research objective guiding the research of this thesis, and it describes the overall research design and context. Then it highlights the main contributions of this thesis, and provides an overview of the papers which contain these contributions.

### 1.1 Problem outline

This thesis is concerned with how agile software development projects can achieve adequate security within acceptable costs – termed “good enough” security in this thesis. How much security would be “good enough” will vary between projects and may even vary with time as



development progresses and requirements are negotiated. It may even change after development, due to new security breaches, new usage scenarios, or product updates.

A basic assumption in this thesis is that to achieve good enough security it is important to make good decisions on the security that is needed. Software development projects are unlikely to get to good enough security without the ability to make priorities on what security is needed throughout the project. This furthermore requires that security is given a certain level of priority in the development project overall. When security decisions are called for, the need for such decisions should be realised and decisions made with sufficient quality, recognising that spending time on prioritisation is also a prioritisation issue. As prevention of software security vulnerabilities is more efficient than detection and response [6], such security prioritisation should happen early on. Still, security prioritisation is not something that can be done once and for all. Rather, it needs to be a continuous endeavour spanning a software product's lifecycle.

Making good priorities on software security requires an ability to navigate the complex landscape of security and software development. It is necessary to understand the various security threats and select mitigation strategies that match the security needs. There is also a need to consider a variety of project limitations such as resources, budget, and time restrictions, as well as potential costs related to qualities such as usability and performance.

In agile software development, software development teams are expected to be self-managed, capable for the job, and motivated, and thus should be trusted to “get the job done” [7]. This has implications for prioritisation of software security. According to the agile principles, software development teams should be trusted also with security, leaving no clear role for a security expert wanting to ensure security is given proper priority throughout. However, literature has documented an overwhelming number of challenges to software security and other quality aspects in agile software development documented in the literature [2–5, 8, 9]. This indicates that simply trusting developers to fix security is not a good option.

This thesis considers security experts in the organisation as one type of actor that can be capable and willing to improve security prioritisation in agile software development projects. Although the agile principles encourage self-managed teams, the principles also point to the importance of giving the development teams a proper environment and support [7], something which is a potential role for a security expert. Even though literature documents many challenges in the relationship between security experts and development projects [10, 11], there is also evidence that, when successful, security experts can increase developers' sense of responsibility for security [12]. Moreover, they can act as triggers for security when time pressure poses challenges for security and its prioritisation [13]. It is therefore important to enable security experts to be successful influencers for software security. An important prerequisite is to have knowledge on what influences an agile software team to prioritise security, as well as what

hinders security in being prioritised. Furthermore, security experts can benefit from evaluated strategies to apply to strengthen the security prioritisation.

Consequently, this thesis is concerned with the larger problem of ongoing security prioritisation in agile software development, considering and studying a broad set of influences on the security priority. However, it has an emphasis on knowledge and techniques that are relevant for security experts who want to influence this priority.

## 1.2 Research objective and design

The objective of this thesis is to support agile development projects in making efficient and effective decisions on the software security they need, in terms of both security requirements and practices. Thus, the thesis addresses the following overall problem:

### **Overall research problem:**

How can regular security prioritisation be integrated into agile software development so that software products end up with a level of security that is “good enough”?

This overall problem calls for design of solutions, e.g., in form of methods, process support, guidelines, or checklists. Thus, this thesis takes a design science approach [14]. Design science aims for improvements in a context, where the context also needs to be understood. It iterates between two problem-solving activities: designing artifacts to bring about improvements, and answering knowledge questions about the context and the artifact in the context. This means that design science involves design problems as well as knowledge questions. This thesis has the following design problem, structured according to the recommendations of Wieringa [14]:

### **Design problem:**

- *Improve* software security prioritisation
- *by* developing process support for making and following up on software security priorities and decisions
- *that satisfies* the needs of agile development projects
- *in order to* support projects in achieving “good enough” security.

This design problem is supported by the following knowledge questions (RQs)<sup>a</sup>:

---

<sup>a</sup>In this thesis, the term *knowledge question* is used, as this is the term used by Wieringa [14]. However, in the community it is more common to use the term *research question*, abbreviated RQ. Thus, to make it easier for readers, the abbreviation RQ is used throughout when referring to these knowledge questions.

**Knowledge questions (RQs):**

**RQ1** What influences the security prioritisation throughout an agile software development project?

**RQ2** How can security roles and activities support an agile software development project in reaching a “good enough” prioritisation of security?

Knowledge from RQ1 is important to understand which influences to look out for and which to strengthen when seeking to improve software security prioritisation. RQ2 gives knowledge on two concrete types of interventions (roles and activities) that software companies may utilise, and thus gives direct input to the design problem. RQ2 is however quite broad, as there are many potential security activities that can support companies in security prioritisation (including training, risk analysis, testing, etc.) and varied ways security roles can be involved (e.g., at the management level (as a CISO), or in the team (as a security champion)). To focus the research, RQ2 was divided into a set of sub-questions:

**RQ2.1** How does the concept of “good enough” security relate to software security strategies?

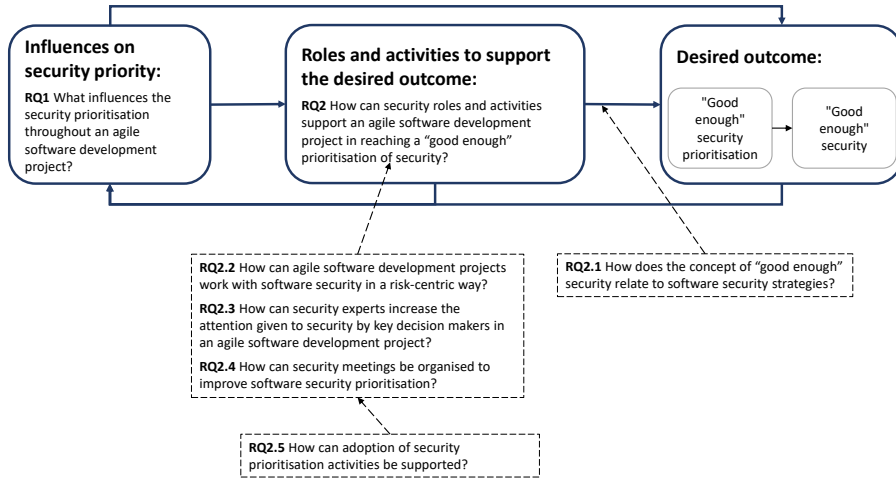
**RQ2.2** How can agile software development projects work with software security in a risk-centric way?

**RQ2.3** How can security experts increase the attention given to security by key decision makers in an agile software development project?

**RQ2.4** How can security meetings be organised to improve software security prioritisation?

**RQ2.5** How can adoption of security prioritisation activities be supported?

Figure 1.1 provides an illustration of the knowledge questions and their relation to each other and the goal of good enough security. As can be seen from this figure, the goal is to achieve good enough security, and in this thesis it is assumed that good enough security is related to good enough security prioritisation. Throughout a project, the security prioritisation can be influenced by a variety of aspects, and understanding these is the focus of RQ1. However, it is also possible to take active action to influence the security prioritisation, as is considered in RQ2. The figure also illustrates that the actions taken (RQ2) can result in changes to influences (RQ1) and that the security prioritisation and the security achieved can feed back to the influences and to the actions. Regarding the sub-questions to RQ2, RQ2.1 is concerned with identifying characteristics of strategies that are likely to support teams in achieving good enough security, while RQ2.2-2.4 are digging deeper on certain strategies that could lead to good enough security. When starting the work on the thesis, one assumption was that risk-centric approaches should be considered (RQ2.2), as these are commonly recommended for information security [15–17] and



**Figure 1.1:** Overview of the knowledge questions

software security [18, 19]. Later on, findings pointed towards the important role of meetings (RQ2.4) and the importance of involving a broad set of actors, including decision makers (RQ2.3). RQ2.5 is concerned with the potential for adoption of the studied strategies.

### 1.3 Research context

Design science puts a strong emphasis on understanding the context and the artefact in the context. Thus, this thesis relies on empirical qualitative studies in software development organisations.

The problem of security prioritisation is assumed to be relevant for a broad set of companies. There is evidence in literature that security requirements and other quality aspects are commonly neglected when using agile software development approaches [2–5]. This points towards security prioritisation being a general problem in agile software development projects. Still, this thesis mainly studies small and medium sized companies. Research points to small and medium sized companies as having a larger potential for improvement in their software security than larger companies [20, 21]. Smaller companies are less likely to have a strong security department and a software security program, something that is necessary in order to be considered mature in relation to software security, e.g., according to the Building Security In Maturity Model (BSIMM) [22]. This does not take away their need for good enough security; a lot of our software is developed by smaller companies. However, smaller companies might need other ways to structure and think about software security initiatives than larger enterprises. Thus, it is important to increase knowledge on how small and medium sized development companies could

be supported in reaching good enough software security.

As already claimed, software security is now important for software in general, not only software considered security critical. The challenge of determining what is good enough is likely to be even more pressing for “normal” software products where security is one of several qualities, and not a key feature of the product. Thus, this work studies security prioritisation in software development projects that do not have security as its main goal.

## 1.4 Main contributions

Four main contributions have been identified from this thesis. In the following we provide a brief overview of each contribution. Contribution C1 responds to knowledge question RQ1, contributions C2-3 contributes to knowledge question RQ2, while C4 contributes to both knowledge questions. The main contributions are explained in more detail in Section 4.

### C1 A conceptual model of the influences on security priority in agile software development

The priority given to security is influenced by the presence of a *driving force* for security, the *visibility* of security, the *motivation*, the *room to manoeuvre*, and the *process match*. These influence categories are based on a longitudinal case study in one company. They are also prominent in the literature and were found useful for understanding the effect of meetings on the priority given to security in studies that are part of this thesis.

### C2 Identified and evaluated strategies that security experts can take in influencing the security priority of agile software development projects

For each of the influence categories from C1, recommendations are provided for security experts on what actions to consider and what challenges to beware of in order to support security prioritisation. These recommendations are based on a study of the initiative of one security expert in one company. Further, based on theory on objectivity from philosophy and qualitative research methods, the following strategies were identified as important for moving towards good enough security: including a variety of perspectives, building interactional expertise, and supporting confirmability.

### C3 A new and evaluated meeting approach for continuous software security in agile software development

This thesis proposes a meeting approach named the Security Intention Meeting Series. With this meeting approach, regular security meetings are arranged where 1) meeting participants

are not primarily security experts but rather include key decision makers in the project, 2) the primary task is to identify and assess security needs, and make prioritisations and decisions on the next steps, and 3) the meeting approach itself is flexible and can be adjusted to the needs of the company when it comes to meeting scheduling and organisation. This meeting approach was instantiated and studied in three companies. This resulted in 19 points of practical advice for agile software development companies wanting to apply this meeting type.

**C4** Rich descriptions of practical experiences with improving software security prioritisation, bridging the gap between science and practice

This thesis provides rich descriptions of practices applied in companies to improve software security prioritisation. This includes a description of a security requirements initiative by one security expert, as well as descriptions of experiences with performing regular security meetings aimed towards making prioritisations and decisions related to security. Such rich descriptions of practical experiences are an important contribution to future research. Within this field, there is a call for more empirical studies with companies [9, 23–26]. To further support future research, the published study results include descriptions of implications for research. To illustrate, nine implications for research are highlighted from the study of the Security Intention Meeting Series with three companies, offering direction for further studies within this topic.

## 1.5 Overview of papers

The contributions are documented in the following research papers. Note that these are numbered according to the chronology of the work rather than the chronology of their publication. All papers are published. The papers are included in their complete form in Appendix A. Section 4 explains how these papers constitute the contributions C1-4.

- A** I. A. Tøndel, M. G. Jaatun, D. S. Cruzes and N. B. Moe, 'Risk Centric Activities in Secure Software Development in Public Organisations,' *International Journal of Secure Software Engineering (IJSSE)*, vol. 8, no. 4, pp. 1–30, 2017. DOI: [10.4018/IJSSE.2017100101](https://doi.org/10.4018/IJSSE.2017100101)
- B** I. A. Tøndel, M. G. Jaatun and D. S. Cruzes, 'IT Security Is From Mars, Software Security Is From Venus,' *IEEE Security & Privacy*, vol. 18, no. 4, pp. 48–54, 2020. DOI: [10.1109/MSEC.2020.2969064](https://doi.org/10.1109/MSEC.2020.2969064)
- C** I. A. Tøndel, M. G. Jaatun, D. S. Cruzes and L. Williams, 'Collaborative security risk estimation in agile software development,' *Information and Computer Security*, vol. 27, pp. 508–535, 4 2019. DOI: [10.1108/ICS-12-2018-0138](https://doi.org/10.1108/ICS-12-2018-0138)

- D** I. A. Tøndel and M. G. Jaatun, ‘Towards a Conceptual Framework for Security Requirements Work in Agile Software Development,’ *International Journal of Systems and Software Security and Protection (IJSSSP)*, vol. 11, no. 1, pp. 33–62, 2020. doi: [10.4018/IJSSSP.2020010103](https://doi.org/10.4018/IJSSSP.2020010103)
- E** I. A. Tøndel, D. S. Cruzes, M. G. Jaatun and K. Rindell, ‘The Security Intention Meeting Series as a way to increase visibility of software security decisions in agile development projects,’ in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, 2019, pp. 1–8. doi: [10.1145/3339252.3340337](https://doi.org/10.1145/3339252.3340337)
- F** I. A. Tøndel, D. S. Cruzes and M. G. Jaatun, ‘Achieving “Good Enough” Software Security: The Role of Objectivity,’ in *Proceedings of the Evaluation and Assessment in Software Engineering*, 2020, pp. 360–365. doi: [10.1145/3383219.3383267](https://doi.org/10.1145/3383219.3383267)
- G** I. A. Tøndel, D. S. Cruzes, M. G. Jaatun and G. Sindre, ‘Influencing the security prioritisation of an agile software development project,’ *Computers & Security*, vol. 118, 2022, issn: 0167-4048. doi: [10.1016/j.cose.2022.102744](https://doi.org/10.1016/j.cose.2022.102744)
- H** I. A. Tøndel and D. S. Cruzes, ‘Continuous software security through security prioritisation meetings,’ *Journal of Systems and Software*, vol. 194, 2022. doi: <https://doi.org/10.1016/j.jss.2022.111477>

In addition to these primary papers that are part of this thesis, there are a number of papers that bear relevance to the work in this thesis. These are listed below, and their relation to the thesis is briefly described in Appendix B.

- I** I. A. Tøndel, M. G. Jaatun, D. Cruzes and T. D. Oyetoan, ‘Understanding Challenges to Adoption of the Protection Poker Software Security Game,’ in *Computer Security. SECPRE CyberICPS 2018*, S. K. Katsikas, F. Cuppens, N. Cuppens, C. Lambrinouidakis, A. Antón, S. Gritzalis, J. Mylopoulos and C. Kalloniatis, Eds., 2019, pp. 153–172. doi: [10.1016/j.cose.2018.09.003](https://doi.org/10.1016/j.cose.2018.09.003)
- J** I. A. Tøndel, T. D. Oyetoan, M. G. Jaatun and D. Cruzes, ‘Understanding Challenges to Adoption of the Microsoft Elevation of Privilege Game,’ in *Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security (HoTSoS ’18)*, 2018. doi: [10.1145/3190619.3190633](https://doi.org/10.1145/3190619.3190633)
- K** D. S. Cruzes, M. G. Jaatun, K. Bernsmed and I. A. Tøndel, ‘Challenges and Experiences with Applying Microsoft Threat Modeling in Agile Development Projects,’ in *2018 25th Australasian Software Engineering Conference (ASWEC)*, 2018, pp. 111–120. doi: [10.1109/ASWEC.2018.00023](https://doi.org/10.1109/ASWEC.2018.00023)
- L** I. A. Tøndel, D. S. Cruzes and M. G. Jaatun, ‘Using Situational and Narrative Analysis for Investigating the Messiness of Software Security,’ in *Proceedings of the 14th ACM /*

*IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'20)*, 2020. doi: [10.1145/3382494.3422162](https://doi.org/10.1145/3382494.3422162)

- M** G. Brataas, I. A. Tøndel, E. Okstad, O. Løkberg, M. G. Jaatun, G. K. Hanssen and T. Myklebust, 'The Quality Triage Method: Quickly Identifying User Stories with Quality Risks,' in *2020 2nd International Conference on Societal Automation (SA)*, 2021, pp. 1–7. doi: [10.1109/SA51175.2021.9507110](https://doi.org/10.1109/SA51175.2021.9507110)
- N** I. A. Tøndel and G. Brataas, 'SecureScale: Exploring Synergies between Security and Scalability in Software Development and Operation,' in *Proceedings of the European Interdisciplinary Cybersecurity Conference (EICC '22)*, 2022, pp. 36–41. doi: [10.1145/3528580.3528587](https://doi.org/10.1145/3528580.3528587)

## **1.6 Structure of the thesis**

This thesis is structured as follows. Chapter 2 introduces the state of knowledge on strategic prioritisation of software security in agile software development, and thus provides a foundation for understanding the motivation for and contribution from this thesis. Chapter 3 describes the design science approach as used in this thesis. Chapter 4 describes the main contributions of this thesis, and how the papers constitute these contributions. Chapter 5 discusses the contribution of this thesis, as well as its limitations, and points to further research avenues. Chapter 6 concludes the thesis. All papers that constitute this thesis are included in their complete form in Appendix A. Secondary papers are summarised in Appendix B.





## BACKGROUND

This chapter introduces the state of knowledge on strategic prioritisation of software security in agile software development, and shows how current literature motivates an emphasis on producing empirical knowledge on practices and experiences in industry. Further, it explains the need for an improved understanding of how security experts can be successful influencers on the priority given to software security in agile development projects, and how this relates to a need for better knowledge on influences on security prioritisation in agile projects. The chapter also explains the importance of meetings in agile practices, something that points to meetings as a potential intervention also for software security.

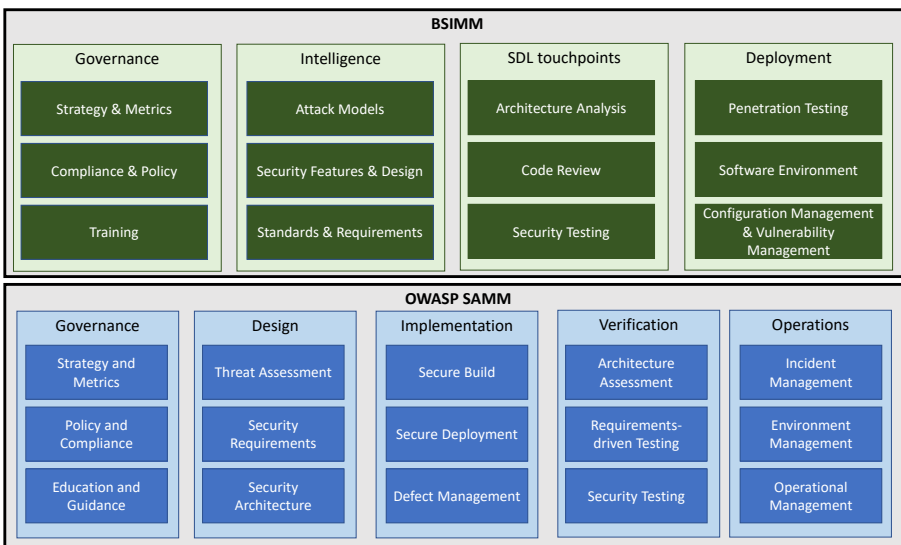
The chapter starts with introducing the role of risk management and security requirements in relation to security prioritisation practices, all central concepts of this thesis. This is followed by an introduction to prioritisation in agile software development, showing how literature supports that agile practices can be challenging for the prioritisation of security. Then it moves on to introducing knowledge on adoption of software security practices, including adoption of security prioritisation, and the potential role of the security expert in supporting adoption.

### **2.1 Risk management, security requirements, and security prioritisation**

Software security is a broad concept, and working with software security can involve varied activities. Software security can be defined as “the practice of building software to be secure

and to function properly under malicious attack" [19]. This practice cannot be confined to a set of features. Rather software security is a quality of a system [41], and it includes the addition of security features as well as building features that are secure.

As security is an overarching concern, it can be hard to grasp and identify in a system and in development practices. It can also involve a broad set of actors [42]. When working with software security there are many potential activities to adopt, from technical features in the code to management practices ensuring or supporting software security is properly addressed. The complexity and totality of the potential activities are illustrated in current maturity models for software security; the Building Security In Maturity Model (BSIMM) [22] and the OWASP Software Assurance Maturity Model (SAMM) [43]. Figure 2.1 gives an overview of the areas covered by these maturity models and illustrate the broadness of the software security work.



**Figure 2.1:** Overview of the areas covered by BSIMM and OWASP SAMM

With the abundance of techniques available for software security (to illustrate, BSIMM12 consists of 112 activities [22]), software security can become quite costly if these are adopted without considering the needs of the projects. Within the field of information security, *risk management* is considered central in ensuring that security activities are aligned with the objectives of the organisation [15]. Within software security, similar arguments are used for risk management activities, e.g., to be more effective in identifying and addressing security vulnerabilities [18] and make better trade-off decisions [19, 44]. There are many approaches and standards related to risk management for information security, including OCTAVE Allegro

[15], ISO/IEC 27005 [16], and the NIST Risk Management Framework [17]. These share many similarities, and based on their content, Paper A identified four key activities involved in risk management: risk analysis, risk treatment decisions, risk treatment follow-up, and risk communication. These types of activities are found also in BSIMM, OWASP SAMM, and other software security initiatives (as outlined in Paper A - see Table 1 in that paper). Many of these activities are however difficult to relate to agile software development in their current form, as there are limited descriptions of how to do this in practice (more on this in Paper C - see Table 1 in that paper)).

A related topic is that of *security requirements*. Literature shows that security requirements are often implicit in agile software development projects [23, 45, 46]. Customers may not have the competence to elicit and follow up on security requirements [45]. However, when security becomes an implicit requirement, this can imply that developers are expected to deliver security without it being clear what this means, and without this being given recognition and being followed up by management [46]. Working with security requirements includes identifying security needs and risks, and documenting what needs to be done regarding security. This should be done in a way that makes security requirements being explicitly communicated to and integrated into the development project. Thus it overlaps substantially with risk managements as explained above. Security requirement initiatives can however run into challenges, as security neither fits neatly into a feature nor a quality view [1, 46]. Further, it is dependent on three distinct aspects: the goals of the stakeholders, the design of the system, and the threats one needs to defend against [41]. Thus, the security requirements cannot be elicited by stakeholders alone, but also need to involve technical resources and people with expertise on security risks [1, 41].

*Security prioritisation* in this thesis, overlaps with both risk management and security requirements work. The concept of security prioritisation includes:

- prioritisation among security requirements and activities
- prioritisation of security vs. other aspects such as functionality, and
- the priority and attention given to security in the day-to-day work.

This makes security prioritisation a broad topic that can encompass many different activities and involve a broad set of roles. Literature documents many challenges to prioritising security and other qualities in agile software development [2–5, 8, 9]. Many of these challenges are not specific for agile development, but are present in more traditional development approaches as well [47]. Challenges include a reliance on individual initiative [9, 23, 48, 49], unclear ownership and responsibility for security [23, 42], and an unclear business case for security [9, 23] (see Paper E for more details on these three challenges).

There is a general call for more empirical studies on software security in agile software development. [9, 23–26]. When it comes to security prioritisation in agile software development,

I am not aware of any studies that examines the priority given to security throughout a development project. Furthermore, the influence of the context is not properly understood, although its importance is well documented; recent studies have shown that challenges can vary considerably between projects [50, 51].

**This thesis is a response to the need for more empirical knowledge on the practices and experiences related to security prioritisation in agile development projects. This includes building knowledge on what influences the security prioritisation (RQ1). It also includes knowledge on how security experts can act and how activities can be used to influence the security prioritisation (RQ2). Among other things, there is a need for more knowledge on how projects can effectively and efficiently work risk-centric with software security in an agile software development context (RQ2.2)).**

## 2.2 Prioritisation in agile software development

Security is one of many risks that have to be managed in software development projects. Agile development projects have been said to treat risk implicitly [52–54], e.g., as part of task prioritisation, and guidance has been characterised as “very general” [55]. Agile software development is based on a set of values and principles outlined in the Agile manifesto [7]. These values and principles can however be practised in different ways, as illustrated by the many agile software development approaches available. Two of the popular ones are Scrum [56] and Kanban [57], and in the following these are used as examples to show how prioritisation is built into these approaches, but in a way that is not necessarily supportive of security getting prioritised.

*Scrum* is a management framework centred around roles, interactions, and artefacts [1]. The *product owner* represents the customer interests and is responsible for setting the direction and priorities regarding what should be worked on, based on the expected business value. The *development team* is responsible for the technical aspects and the development work, while the Scrum process itself is facilitated and supported by a *Scrum master*. Thus, there is a division of responsibilities among these roles, at the same time as their interaction is supported by a set of meetings [1, 56]. A key artefact is the *product backlog* which holds a prioritised set of work items. Two tasks that are strongly linked to prioritisation are the *product backlog refinement* that is concerned with prioritising the items in the product backlog and refining (or detailing) the highest priority items to make them clear for implementation by the development team, and the *sprint backlog creation* that is concerned with deciding upon the objectives of the next iteration (or sprint). According to Scrum, the product owner is responsible for prioritising the items in the backlog, while refinement as well as sprint backlog creation should be done collaboratively by

the team, the scrum master, and the product owner - although in practice it may be done by the product owner or another individual [58]. Sprint backlog creation happens in the sprint planning meeting. Then there are daily stand-up meetings that ensure follow up on tasks. After each sprint, a sprint review meeting is held that includes the client, and there is a sprint retrospective [56, 58].

*Kanban* is concerned with managing workflow and improve performance [57, 59]. It is centred on a Kanban board that visualises the workflow [57, 60]. Key practices are to limit work in progress, visualise the flow of work items, and measure the flow [57]. Compared to Scrum, Kanban is less prescriptive [57] and it does not specify special roles and meetings, although meetings certainly have a place [57, 59, 60].

Looking at both Scrum and Kanban, one can see that prioritisation is built in, and that it involves the team and the product owner (and possibly other actors), and is often done in meetings. However, security or other quality issues have no clear and visible place, and there is no built-in mechanism to ensure security is considered and that security experts are consulted if needed. This is illustrated in the following quote on the relation between software security and Scrum:

There are “subtle mismatches between Scrum’s strengths and the peculiarities of security as a class of requirements. For example, development teams cannot silently take care of security without product management or other stakeholders explicitly motivating and requiring it, but security may remain invisible and intangible for those.” [1]

When it comes to prioritisation within agile software development [61], the most important prioritisation criterion has been found to be business value. Size, effort, and cost estimations are also important inputs, together with project constraints like release dates, budget, and available resources [61]. The business value of security is however often unclear [23], as it is concerned with preventing future damage that may or may not occur [1].

Agile software development values “Responding to change over following a plan” [7]. When comparing with more traditional waterfall development approaches, agile’s approach to requirements have been characterised as “far more temporal, interactive, and just in time” [62]. Requirements are no longer fixed but can be dropped or changed based on new needs or to meet cost and schedule constraints [62]. Within such an approach, traditional ways to deal with security can be troublesome, as illustrated by challenges identified in a review by Oueslati et al. [3]: “Refactoring practice breaks security constraints”, “Changes of requirements and design breaks system security requirements”, “Continuous code changes makes completing the assurance activities difficult”, and “Requirement changes makes the trace of the requirements to security objectives difficult” [3]. At the same time, agile development’s support for change could

be beneficial in addressing changing security threats. Since the review by Oueslati was performed in 2015, there has been a growing understanding of the possibility to successfully merge security with agile practices, e.g., declaring the incompatibility of security and agile development to be a myth [63]. Still, literature identifies numerous challenges to software security and other quality aspects in agile software development [2–5, 8, 9], and documents a common neglect of security in agile development projects [2–5]. Going back to the maturity models of BSIMM [22] and OWASP SAMM [43], there seems to be a mismatch between the kind of strategic and comprehensive software security approaches preached in these frameworks and the challenges to software security in agile software development documented in literature. This is the case despite BSIMM being descriptive and based on practices identified in software companies [22]. This leads to the assumptions that many agile software development organisations have challenges in working strategically with software security, and would not be considered mature according to BSIMM and OWASP SAMM.

Software security has seen a shift from prescriptive approaches, with prescribed changes to the development lifecycle, towards more flexible approaches (e.g., maturity models) that allow for practices emerging from and being adjusted to the needs of the companies [20]. This is more in line with the agile principles encouraging self-organising teams and trust in their ability to “get the job done” [7]. However, the Agile manifesto also hints at the importance of the context and the support it offers to the teams:

“Build projects around motivated individuals.  
Give them the environment and support they need,  
and trust them to get the job done.” [7]

Going back to security prioritisation, trust in the team is built into the agile principles, but such trust is not necessarily blind and should not be viewed as giving the team the full responsibility. Current literature does, however, not give any clear direction as to what is “the environment and support” needed to make security happen in an agile software development setting. Thus, there is a need to understand how to support emerging practices and self-management for security. Furthermore, there is a need to understand how to balance team autonomy with the need to ensure security is included also in cases where the team does not have individuals with a strong awareness of security, to address the limited visibility of security in prioritisation processes in agile development projects.

There are a few techniques available that are targeted towards supporting agile software development in their security prioritisation. An example is Protection Poker [64, 65], which is a collaborative risk estimation game that involves the full team in security prioritisation. Another is a framework that, based on a manual risk assessment, offers automated support for prioritising security requirements that can be added to the product backlog [66]. Further, a study

of incremental and distributed risk management in agile software development has reported promising results [67]. Literature also points to strategies such as stakeholder involvement [1, 42, 68], security reflection [1], co-creation and situated learning [1, 69, 70], and documentation [1](see Paper H for more details).

**This thesis is a response to the need for strategies to improve software security prioritisation in agile software development (the design goal). To get to such strategies, there is a need to better understand how projects can know that their security prioritisation is good, and likely to lead to good enough security (RQ2.1). Meetings already play a central role in prioritisation within agile software development approaches. They should be considered also for software security prioritisation (RQ2.4). Furthermore, getting key decision makers, such as the product owners, onboard, is likely to be important (RQ2.3).**

### **2.3 Security experts and their role in adoption of software security practices**

Security prioritisation is also about adoption; adoption of security prioritisation, and adoption of security practices. The literature provides some insight into what is important for adoption of software security practices. Kanniah and Mahrin [48] performed a literature review to identify commonly cited factors impacting the successful implementation of secure software development practices. The broad set of factors identified include the institutional context, the people involved and their actions, the project content, and the system development process. In a follow-up study with eight experts [49], the following factors were identified as the ten most important ones: 1) security experts, 2) security documentation, 3) project management, 4) developers, 5) project team, 6) security audit team, 7) team collaboration, 8) development time, 9) policy enforcement, and 10) top management. Many of these factors are concerned with the individuals and roles involved and their relations. The importance of the relation between security experts and developers for adoption of security practices are supported in other studies. An example is the study by Xiao et al. [12] on the adoption of secure development tools, that 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” [12].

Commonly reported reasons for not adopting a secure software development lifecycle have been found to be concerns about it being time-consuming and requiring too many resources [71]. This is the case not only for development projects using an agile development approach. However, agile development’s emphasis on delivering “working software frequently” [7] represents a push towards visible features that can be a reason for time pressure in agile projects, impacting the



adoption of software security practices. Research on time pressure shows the importance of security roles to counteract the effect of time pressure. Chowdhury et al. [13] suggested that all organisational units should have a strong relationship with the security department as a way of dealing with the implications of time pressure on security.

The evidence for the importance of the interplay between security experts and developers has been strengthened in the literature throughout the course of the work on this thesis, with a recent publication based on statistics on BSIMM finding a strong correlation between the number of activities adopted and the number of security specialists involved [72]. This points to the security expert as a possible change agent for security, despite many challenges in establishing good relations between security experts and development projects [10, 11]; communication between security experts and developers can be challenging [10, 11] and there can be unclear responsibilities between these groups when it comes to security [11, 42]. It has even been stated that it is better to teach developers about security than to turn IT security engineers into software security people [22]. To support practitioners in navigating these challenges, there is a need for a better understanding of the effect of potential strategies that security experts can utilise to strengthen security prioritisation in agile software development projects. Empirical knowledge on what makes security initiatives from security experts be successful (or not) in an agile development setting, can provide a basis for practical advice for security experts wanting to act as a change agent for security in an agile development setting.

**This thesis studies initiatives by security experts within an agile software development context, to arrive at recommendations directed towards security experts wanting to influence the security prioritisation in agile software development projects (RQ2). Furthermore, this thesis builds knowledge on adoption of the studied security prioritisation strategies (RQ2.5).**

## RESEARCH APPROACH

As already explained in the introduction, this thesis utilises design science as its main research approach. Wieringa defines design science in the following way:

“**Design science** is the design and investigation of artifacts in context. The artifacts we study are designed to interact with a problem context in order to improve something in that context.” [14]

Within software engineering and information systems, there are several design science approaches available. Prominent examples are Hevner et al. [73, 74] and Wieringa [14]. This chapter first gives a brief introduction to the design cycle as described by Wieringa [14] and explains the choice to build on his approach. Then it explains how the design science methodology was applied in the work with this thesis.

### 3.1 The design cycle

Within design science [14], an artefact is a broad concept including anything that can be designed by a researcher, e.g., a software system, a business process, a method, or a technique. The context, on the other hand, is given to the researcher and needs to be investigated and understood by the researcher, but not changed. For the design science researcher both are essential:

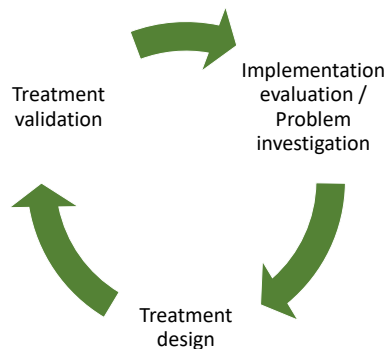
“(...) the artifact itself does not solve any problem. It is the *interaction* between the artifact and a problem context that contributes to solving a problem. An artifact may interact differently with different problem contexts and hence solve different problems in different contexts. It may even contribute to stakeholder goals in one

context but create obstacles to goal achievements in another context. The design researcher should therefore study the interaction between artifacts and contexts rather than artifacts alone or contexts alone.” [14]

Thus, design science is about both design and investigation; it seeks to address both design problems and knowledge questions. In the design cycle, the tasks of design and investigation are iterated over. The design cycle is illustrated in Figure 3.1. Its main steps are:

- *Problem investigation:* Happens before the design of an artefact. The goal is to “identify, describe, explain and evaluate the problem to be treated” [14].
- *Implementation evaluation:* Here the research goal is to evaluate an implementation of a treatment that has been applied in a problem context. The goal does not have to be to prepare for further improvement. It can also be to “describe, explain and evaluate the effects of a past improvement” [14].
- *Treatment design:* Concerns specification of requirements for the treatment and designing the treatment.
- *Treatment validation:* Implies studying “the interaction between an artifact and its context by studying a model of it” [14]. The goal is to develop a design theory that “allows us to predict what would happen if the artifact were transferred to its intended problem context” [14]. Note that ‘model’ here can take various forms, including using action research to investigate the effect of one instance of the treatment in a real-world case, and then use this case as a model for other real-world cases.

Note that this design cycle is part of a bigger engineering cycle with an added step, *treatment implementation*, after treatment validation. This step is concerned with “the application of the treatment to the original problem context” [14], e.g., transferring the product to the market. This is however outside of the scope of a design science research project [14].



**Figure 3.1:** The design cycle (adapted from Wieringa [14])

Comparing the design science approach as described by Wieringa [14] to that of Hevner et al. [73, 74], there are many similarities: both are about creating artefacts to solve real world problems, both iterate over designing and validating the artefact in a design cycle, both emphasise the importance of understanding the problem context, and both calls for new knowledge in relation to the existing knowledge base. Hevner operates with three research cycles: the *relevance cycle*, the *design cycle*, and the *rigor cycle*. The design cycle iterates over activities concerning designing/building artefacts and evaluate them. The relevance cycle considers the relevance of the artefact to the environment, e.g., through identifying requirements and doing field testing. The rigor cycle connects the research with the scientific knowledge base. The choice to base the research approach of this thesis on Wieringa was pragmatic; the design science approaches of Wieringa and Hevner both had their merits and was in line with the goals and needs of the thesis, but I found that the design cycle as described by Wieringa matched well with the approach we wanted to take in this work as it included problem investigation activities into the design cycle.

### **3.2 The design science approach as applied in this work**

Through the work in this thesis, I wanted to be able to offer process support (the artefact) to improve software security prioritisation in agile software development projects. Thus, it was important to understand the context of agile software development projects as well as the potential of process support to improve software security prioritisation. To achieve this, I iterated over the design cycle. Initially the goal was to build understanding, and thus broadness was emphasised over depth. Then the work narrowed down on one company and one way to help this company with a regular security meeting approach. Finally, similar meetings were studied in two more companies to get a broader understanding of how this treatment could be relevant also for other companies. Figure 3.2 outlines the iterations over the design cycle. There the approach is roughly divided into five iterations, where not all steps were addressed in each iteration.

The approach started with an empirical investigation of the problem and the context. This was done through studying the practices and challenges of public companies related to software security, and through studying one prominent technique from literature, the Protection Poker software security game [64, 65], with students. At this stage, the context models studied (public companies and students) were selected based on convenience, as the goal was to get started in understanding the context, its challenges, and experiment with potential treatments. This initial empirical investigation gave a better understanding of the challenges this work needed to address, both in the context and with techniques. This improved understanding was used to guide the work in the second cycle, where I identified and structured relevant empirical findings documented in literature, and developed an initial treatment design (the Security Intention

Meeting Series). Both these outputs were important in sharpening the research focus in the remaining cycles.

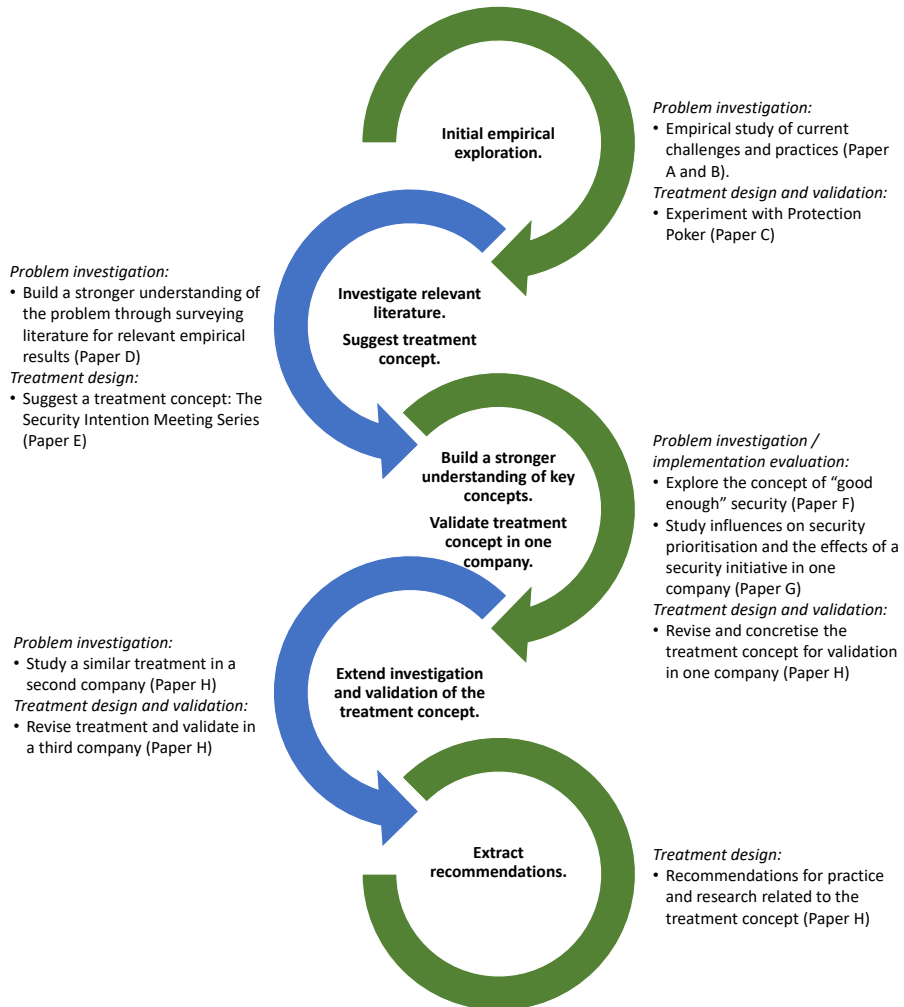
After having established this broad view of the problem, I now dug deep into the specifics of security prioritisation in one company, investigating the influences on security priority in two projects in this company. I also investigated more thoroughly the concept of “good enough” security, using my experiences from this company together with theory on objectivity and experiences from co-authors. Both these activities were useful to come to a deeper understanding of what would be required from a treatment, and how to evaluate a treatment. Then, a revised treatment design was brought to this company, based on the Security Intention Meeting Series approach outlined in a previous iteration. After evaluating this treatment in the company, two other companies were brought into the study, allowing for studying the same or a similar treatment in different contexts. Finally, the results of the studies of instantiations of the Security Intention Meeting Series were made into recommendations for research and practice on using regular security prioritisation meetings. Note that although the approach taken was that of design science, and the initial goal was to design and validate a treatment, it became evident along the way that prescriptive approaches were likely to not reach broad adoption. Thus, the output of this thesis in terms of treatment, is in form of recommendations and not a prescribed technique to apply.

As outlined in the Introduction, I explored two main knowledge questions in this work. These were relatively stable throughout the work with the thesis, while the sub-questions to the second knowledge questions varied throughout. Figure 3.3 places the papers and their research questions in relation to the overall knowledge questions and the design cycle iterations. For RQ1, the trajectory of the investigation moves from more general studies to a longitudinal case study, all culminating in a conceptual model of influences on security priority. For RQ2, the initial emphasis was on understanding how to work risk-centric (RQ2.2) with software security, either generally or through meetings (using the technique Protection Poker [64, 65]), and why this was not much adopted among the studied public organisations (RQ2.5). The results from this investigating prompted an exploration of the role of security experts (RQ2.3), a topic that was followed up in later studies as well. The concept of good enough security (RQ2.1), a concept central to this thesis, was explored. Then the emphasis moved towards the role of meetings (RQ2.4), identifying a potential meeting concept (the Security Intention Meeting Series) and studying meetings in companies (including their adoption (RQ2.5)).

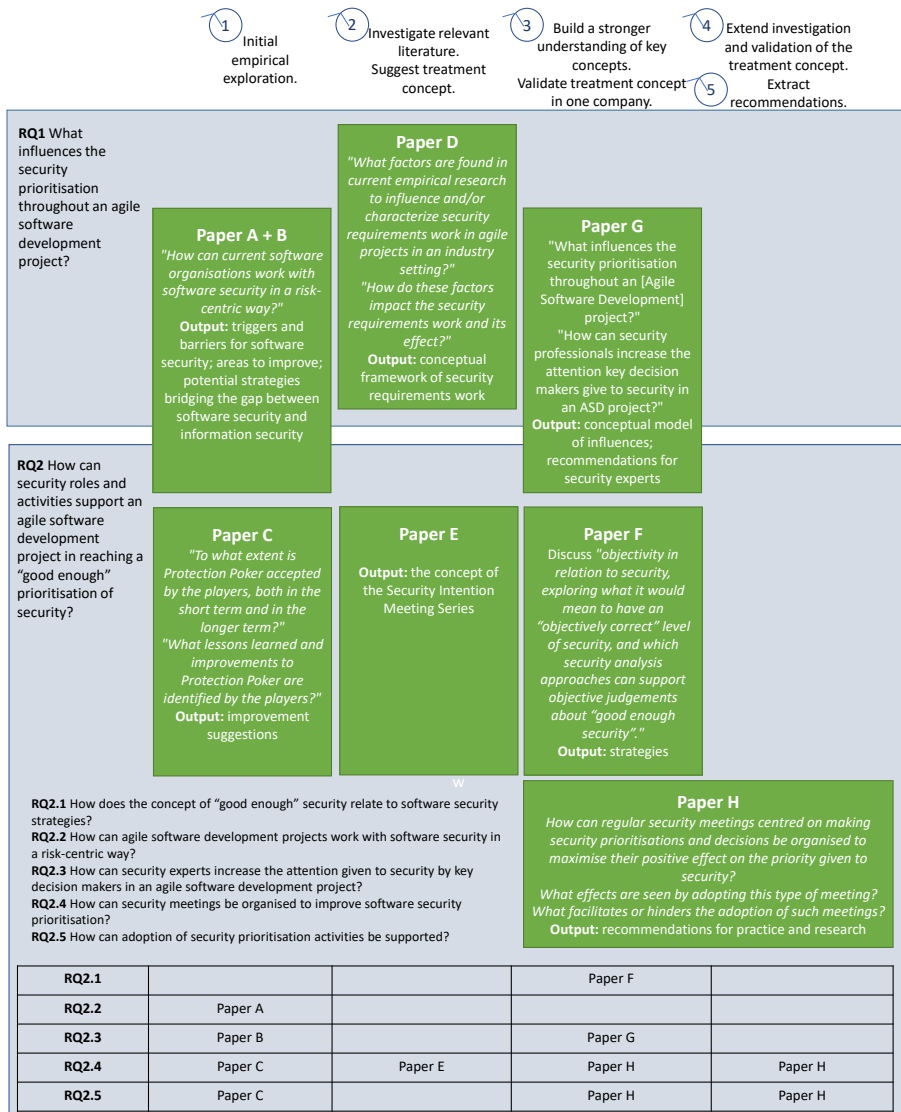
### **3.3 Use of case studies and technical action research within a design science approach**

The design science approach can encompass a variation of research methods. Wieringa [14] outlines the following example research methods that can be used with design science: observational case studies, single-case mechanism experiments, technical action research, statistical difference-making experiments, and survey research. The problem that this thesis investigates (i.e., security prioritisation in agile software development projects) was considered difficult to move to the laboratory, as it represents a complex social phenomenon where the boundaries between the phenomenon and the context was blurred [75]. Further, there was not a strong theoretical basis to build on for survey research. Thus, we opted for the alternatives that were closest to the field: observational case studies and technical action research.

Wieringa describes observational case studies as “a study of a real-world case without performing an intervention” and technical action research as “the use of an experimental artifact to help a client and to learn about its effects in practice” [14]. In this thesis, both were necessary. Case studies were used to understand the context and treatments already implemented, while action research was used when introducing treatments in the context to help the companies and evaluate the treatment. Table 3.1 gives an overview of the use of case studies and action research in this thesis. Note that in the initial studies (as reported on in Paper A, B, and C), other research methods were utilised as well, and thus there are aspects of those studies that would not be consider neither case study research nor action research. Furthermore, a literature review was performed to synthesise findings from related empirical studies. For all the studies, the research methods and the rationale for selecting these objects of study are described in the papers.



**Figure 3.2:** The design cycle iterations of this thesis. Note that for practical purposes, to distinguish the iterations in the figure, I have used two colours and the arrows representing the cycles go in different directions. This does not have any significance when it comes to the method - each arrow represents one design cycle iteration.



**Figure 3.3:** The relation between the knowledge questions, the research questions in the papers, and the design cycle iterations. Some papers have contributions to more than one knowledge question.



**Table 3.1:** Overview of studies (CS = case study; AR = action research; LR = literature review, I = interviews)

Study	Paper	Method	Treatment	Context	Rationale
S1: Risk-based practices in public companies	A, B, F	CS, I	NA	Public companies	Used data collected in previous studies to understand the problem context
S2: Capstone study	C	AR	Protection Poker	Students	Experiment with one promising technique from literature. Easy access to many cases to study.
S3: Literature review	D	LR	NA	Published empirical studies in companies	Synthesise findings from related studies to understand the current state of knowledge and identify research needs.
S4: Influences and the security requirements initiative	F, G	CS	Security requirements initiative	One medium-sized company	Deeper understanding of what influences the priority given to security, including the role of the security expert.
S5: Security meetings	H	AR	Security Intention Meeting Series instantiation	Two companies (one medium-sized, one small)	Instantiation of the security intention meeting approach in two companies, to help them and evaluate the approach.
S6: Security group	H	CS	Security intention meeting instantiation	One medium-sized company	Company already running meetings with similar goals as the security intention meeting approach.

## CONTRIBUTIONS

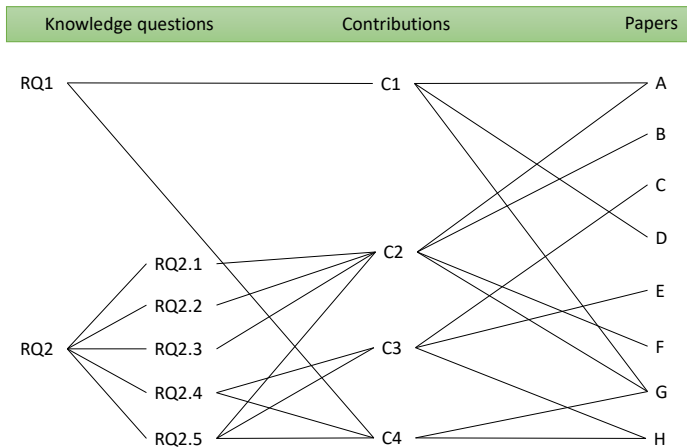
This chapter gives an overview of the main contributions of this thesis. For each main contribution, a short summary of the contribution is provided. Then the trajectory towards the contribution is described, including how it relates to the individual papers, and an argument is provided as to why this constitutes a contribution to the current body of knowledge. More details about the contributions and how they advance state of the art can be found in the papers. Figure 4.1 shows the relations between the papers, the contributions, and the knowledge questions. Table 4.1 summarises the contributions of the papers.

### 4.1 C1 - A conceptual model of the influences on security priority in agile software development

**Main contribution:** A conceptual model of influences on security priority. The model consists of the following influence areas: driving force, visibility, motivation, room to manoeuvre, and process match.

**Main foundation:** A longitudinal case study of influences on the priority given to security in two software development projects.

**Purpose:** Identify areas to focus on for practitioners that want to influence the priority given to security. Provide a foundation for researchers wanting to investigate how companies can be supported in organising their software security work to support security being given priority by self-managed teams.

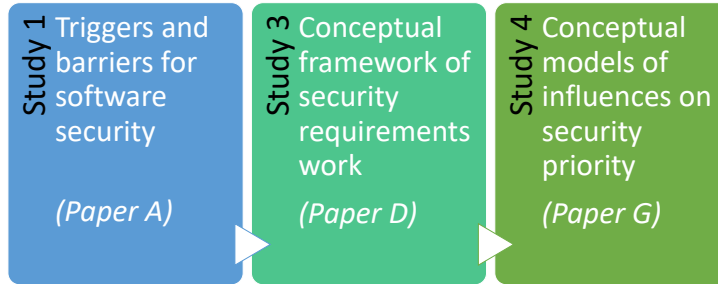


**Figure 4.1:** Illustration of the relation between the knowledge questions, the contributions, and the papers

Understanding “what influences the security prioritisation throughout an agile software development project” (RQ1) implies understanding how security is given priority (or not) in agile software development, how it changes throughout a project, and what leads to this change. This is a very broad topic, as prioritisation in agile software development happens throughout and many stakeholders and aspects has been known to be influence or be actively involved or considered in this prioritisation [41, 61]. In addressing this knowledge question, this thesis thus sought to gain a broad overview, while at the same time dig deep into one case to ensure a thorough understanding of what influences can play out in a project. This led to the following main contribution: a conceptual model of influences on security prioritisation, mainly based on a deep-dive into two projects in one company. Figure 4.2 gives an overview of the trajectory towards this contribution.

The conceptual model of influences on security priority is mainly based on Paper G, but also bears relations to Paper A and Paper D:

- *Paper A contribution:* The trajectory towards this contribution started in Paper A, where a study of public companies revealed that the companies we studied did not seem to follow a risk-based approach to software security to any large extent. Rather, other aspects seemed to have a stronger impact on the priority software security was given. Based on the study, we identified triggers and barriers for software security activity.
- *Paper D contribution:* In Paper D, we used published empirical studies to create a



**Figure 4.2:** The progress towards contribution C1

conceptual framework for security requirements work in agile software engineering. The motivation was to build an understanding of the key concepts that impact and/or characterise security requirement work, which is central to software security prioritisation. This paper builds a foundation for research within the area of security requirements work, by structuring the empirical knowledge and identifying knowledge gaps. In this thesis, this paper became a basis for the work presented in Paper G, providing research direction and an initial coding structure.

- *Paper G contribution:* In Paper G we studied one company and their initiative to improve software security. We followed two projects, and identified influences on the priority given to security. These influences were organised into a conceptual model. Paper G defined the influence categories of the conceptual model, described how the influences from the case study were grouped into influence categories, and showed how these influence categories are well supported in literature. As part of this, Paper G, mapped these influence categories back to the triggers and barriers identified in Paper A, showing that these triggers could fit within the conceptual model.

The conceptual model of influences on security priority represents a contribution to the knowledge in the field for the following reasons. Although the literature documents numerous challenges to software security and other qualities in agile software development [2–5, 8, 9] and points to the risk of software security being neglected in agile development projects [2–5], there is to my knowledge no studies going in depth on one project to understand what influences security priority throughout a project. Thus, this represents new knowledge despite the influence areas being well documented in related studies.

The usefulness of the conceptual model for research is demonstrated by its usefulness in the analysis performed in Paper H, where these influence areas were used to build understanding of the effects of security meetings in that study. Furthermore, the broad support for the influence

categories in literature supports the assumption that this conceptual model can be useful for a broader set of projects, although being based on findings from one company. We expect that for security experts that want to support self-managed development teams in their software security prioritisation, knowledge of these influence categories can help them assess the situation, identify areas of improvement, and provide more targeted support for the software security prioritisation in the teams. This, however, needs to be evaluated in future studies.

## 4.2 C2 - Identified and evaluated strategies that security experts can take in influencing the security priority of agile software development projects

**Main contribution:** Recommendations for security experts on how to support prioritisation of security in form of: 1) strategies that are important for moving towards good enough security (including a variety of perspectives, building interactional expertise, and supporting confirmability), and 2) recommendations for what to consider doing and what to beware of doing in order to support security being prioritised by a project.

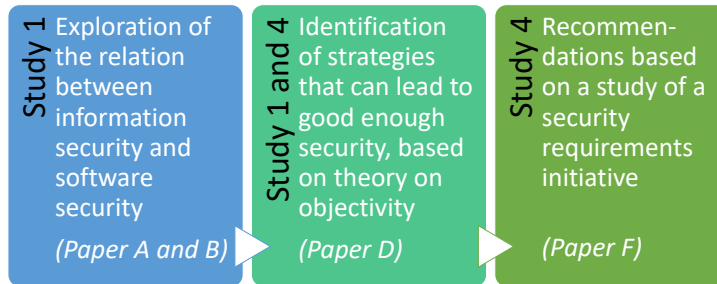
**Main foundation:** 1) Theory on objectivity from philosophy and qualitative research methods, and 2) a longitudinal case study of a security requirements initiative led by one security expert in a company.

**Purpose:** Provide recommendations for practitioners that want to increase the priority given to security in agile software development projects. Extend knowledge on the security expert role in relation to agile software development teams.

To understand how security experts can support an agile software development project in reaching a good enough prioritisation of security (RQ2), it is necessary to understand both what can make this interaction challenging and what are potential effects of certain strategies that a security expert might take. In this thesis, the initial emphasis was on identifying challenges to this interaction and exploring reasons why the relation between information security and software security could be challenging. Then the work moved towards exploring potential strategies, either from a theoretical or a practical perspective. Figure 4.2 gives an overview of the trajectory towards this contribution.

The strategies for security experts are mainly based on Paper F and G, but also bear relations to Paper A and B:

- *Paper A and B contribution:* Study S1 revealed challenges related to responsibilities and stakeholder collaboration that included a disconnect between those responsible for information security and those involved in software development. These challenges



**Figure 4.3:** The progress towards contribution C2

were identified and discussed in Paper A. Paper B then extended the discussion of this relation, based on previous work identifying and discussing similar challenges [76], thus contributing to a better understanding of why this relation can turn out to be difficult. Potential strategies that were suggested, included to have Security Champions in the teams and engage the team in risk-centred activities.

- *Paper F contribution:* A central concept in this thesis is that of “good enough” security. Paper F explored this concept using theory on objectivity from philosophy and qualitative research methods literature. It used this literature together with experiences from Studies S1 and S4, as well as previous experiences from the authors, to explore the concept of good enough security and arrive at three strategies likely to be important in supporting good enough security, namely including a variety of perspectives, building interactional expertise, and supporting confirmability.
- *Paper G contribution:* Paper G reported on a longitudinal case study that followed a security requirements initiative led by a security expert in a company. Based on the security expert’s experiences and the conceptual model of influences on security priority (C1), the paper provided recommendations for what security experts should consider doing and what challenges they should beware of, to support security gaining priority in an agile software development project.

The recommendations made in terms of strategies for security experts, represent a contribution to the knowledge in the field for the following reasons. Although challenges in the relation between information security and software security is well documented in literature [10, 11, 76], there are not many studies focusing on this relation. Thus, the reported experiences in this thesis, especially in Paper G that report on one initiative in detail, contributes to improved understanding of the experiences of security experts. I am not aware of other studies that follow a similar security initiative to provide recommendations for security experts in strengthening security prioritisation in agile development projects. The concept of good enough security is present in

literature [77–79]. Still, using theory on objectivity with the aim to explore this concept and arrive at potential strategies, represents a new approach to understanding this important concept.

This contribution is mainly directed towards practitioners. The recommendations are rooted in practical experiences in companies. Still, more research is needed to know if these recommendations are relevant for a broader set of contexts than what was studied in this thesis. For researchers, this contribution contributes to a better understanding of the security expert role in software development, thus being a foundation for developing improved support for security experts wanting to interact with development projects to improve the software security.

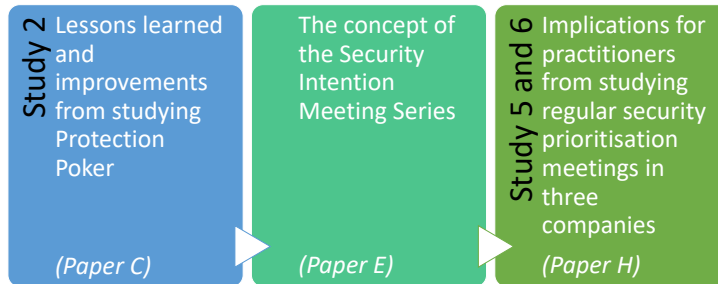
### 4.3 C3 - A new and evaluated meeting approach for continuous software security in agile software development

**Main contribution:** The description of a meeting concept, *the Security Intention Meeting Series* and an evaluation of three instantiations of this meeting concept, resulting in 19 implications for practice.

**Main foundation:** Case study research and action research with companies.

**Purpose:** Provide practical advice for agile software development companies wanting to use meetings to improve their security prioritisation. Build knowledge on the potential effect of security meetings on security prioritisation.

Several activities could have the potential to support an agile software development project in reaching good enough prioritisation of security (RQ2). Examples of activities that could support this includes security testing, risk analysis, and software maturity assessments using, e.g., BSIMM (as discussed in Paper F). The emphasis on security meetings in this thesis builds on results from Study S1 (Paper A and B). Study S1 pointed towards the need to: 1) address challenges related to responsibilities and stakeholder cooperation, 2) build risk perception and competence, and 3) develop practical ways of doing risk analysis that matches with the needs of agile development projects. Protection Poker [64, 65] was identified as a potential technique to adopt to achieve this as it: 1) was able to gather a broad set of stakeholders, 2) had already been found to be able to raise awareness and knowledge on security, and 3) offered a practical risk estimation technique specifically suited for agile development. Thus, it was decided to study the adoption and effect on this technique (Study S2). Findings from the study of Protection Poker led to the development of a new meeting concept, the Security Intention Meeting Series, that was instantiated and evaluated in three companies (Study S5 and S6). This led to 19 recommendations for practitioners wanting to use security decision meetings to improve security prioritisation. The trajectory towards this contribution is illustrated in Figure 4.4.



**Figure 4.4:** The progress towards contribution C3

The recommendations for practitioners on how to use security decision meetings are mainly based on Paper E and H, but also bear relations to Paper C:

- *Paper C contribution:* Protection Poker was studied with students and with industry practitioners. This led to the identification of eight benefits and 17 challenges with the technique. Many of the challenges experienced with Protection Poker had, however, also been identified for other security activities, implying that they are general challenges not related to this particular technique. The study identified four issues that needed to be improved for Protection Poker: making the time needed to play Protection Poker more acceptable for the teams; ensuring impact from playing Protection Poker on the security of the end-product; better integrate Protection Poker with project planning activities, and; clarify the scenarios for better adoption of Protection Poker in a project. Suggestions were made for how to improve upon all these concerns.
- *Paper E contribution:* Paper E proposed a new meeting concept called *the Security Intention Meeting Series*. This meeting concept was developed in response to evidence from literature that security is often dealt with in an “accidental” way in agile software development projects, being neglected or left to the interests of individuals involved in the project [2, 3, 23]. The meeting concept was also a response to one of the main challenges identified for Protection Poker, namely finding the time to play Protection Poker in every iteration. This proposed security meeting concept is flexible in how it is organised, including its timing, and is intended to involve key decision makers as participants.
- *Paper H contribution:* The Security Intention Meeting Series proposed in Paper E, was instantiated in two companies in Study S5. Furthermore, a similar meeting approach was studied in another company already running a regular security group meeting (Study S6). Paper H reports the results from studying these three meeting instantiations, describes several positive effects on security prioritisation, and highlights 19 implications for practice. Furthermore, the paper relates the findings to literature that has reported experiences from



related techniques, including to findings from the study of Protection Poker (Paper C).

These recommendations for practitioners represent a contribution for the following reasons. Although meetings are common within agile software development, there is limited research on their effectiveness when it comes to software security. There are a few studies available on relatable meeting types (see Paper H for an overview) but these have main differences with the Security Intention Meeting Series approach. The suggestions for this meeting approach fills a gap, and provides practitioners with another approach to consider for their projects. Furthermore, the insight provided into the effect of this meeting type, as well as the improved understanding of strategies that can support this effect, largely represent new knowledge.

When it comes to usefulness, the emphasis on interactions and face-to-face discussions found within agile software development [7] supports the use of meetings. Thus, there is reason to assume that security meetings are relatively easy to integrate into agile development practices. The studies of security meetings that are part of this thesis (Study S2, S5, and S6) have demonstrated the potential usefulness of such meetings, while showing that adoption still can be challenging. This contribution can help companies apply a meeting approach that is more likely to be applied and be effective, as they can build on the experiences in the studies that are part of this thesis. For researchers, knowledge on effects of such meetings, as well as insight into what brings about this effect, is useful to understand the role of meetings in supporting security prioritisation in agile software development projects. Such knowledge can be built upon in future research, to get to a more complete understanding and better advice to practitioners.

#### **4.4 C4 - Rich descriptions of practical experiences with improving software security prioritisation, bridging the gap between science and practice**

**Main contribution:** Rich descriptions of practices and experiences from companies, and identification of their research implications.

**Main foundation:** Case study research and action research with companies

**Purpose:** Support further research on this topic, giving insight into the practical challenges experienced in companies and identifying important research needs moving forward.

The case studies and action research studies in companies were aimed to arrive at process support for companies, and thereby improve their software security prioritisation. However, these studies also provide an important contribution to research in that it provides a deep insight into the practical experiences and challenges of companies in doing security prioritisation and in applying meetings to improve their security prioritisation.

The knowledge transfer to research is mainly based on Paper G and H:

- *Paper G contribution:* The study of the security requirements initiative initiated by the security officer of the studied company gave insight into the practicalities of doing security prioritisation, and of trying to influence security prioritisation. Furthermore, it described how a varied set of influences on the security priority played out during the course of the studied projects. Research needs were identified based on this study.
- *Paper H contribution:* This paper provided a rich description of the experiences of the companies that applied the meetings, and identified nine implications for research.

This represents a contribution to research for the following reason. There is a call for more empirical research within this topic, as few detailed descriptions of experiences and practices of companies exists in the research literature [9, 23–26]. These papers report on empirical studies, and do so with rich descriptions of experiences.

This contribution is mainly aimed towards researchers, and useful to support future research. It is intended to build a stronger understanding of the practical challenges experienced by companies, as input to theorisation and to guide development of new approaches and recommendations aimed towards practitioners. Still, practitioners can also find it useful get access to rich descriptions of experiences in other companies to learn from these.

**Table 4.1:** Overview of paper contributions

<b>Paper</b>	<b>Relevance to the thesis</b>	<b>My contribution</b>
A	Found that companies generally did not work risk based with software security, but there were other triggers and barriers for software security (C1). Identified challenges related to responsibilities and stakeholder collaboration, including a disconnect between those responsible for information security and those involved in software development (C2).	I was the lead author and wrote about 90% of the manuscript. I was strongly involved in data collection in one of the three sub-studies, and responsible for analysing the combined data from all the three sub-studies.
B	Extended upon the discussion of the relation between information security and software security, started in Paper A, and identified potential strategies moving forward (C2)	I was the lead author and wrote about 80% of the manuscript. I was strongly involved in data collection in one of the three sub-studies, and responsible for analysing the combined data from all the three sub-studies.
C	Studied one meeting technique, Protection Poker, identifying benefits, challenges, and improvements (C3).	I was the lead author and wrote about 90% of the manuscript. I was responsible for organising the capstone study. I performed all data analysis.
D	Developed a conceptual framework for security requirements work in agile software engineering based on published empirical studies (C1).	I was the lead author and wrote close to 100 % of the manuscript. I identified and analysed the primary and secondary studies, and synthesised them into a conceptual framework.
E	Conceptualised a meeting technique, the Security Intention Meeting Series, intended to involve key decision makers in regular security prioritisation (C3).	I was the lead author and wrote close to 100% of the paper. I had the idea for this meeting, which was refined in discussions with the co-authors.
F	Explored the concept of “good enough” security based on theory on objectivity, resulting in identification of three strategies (C2).	I was the lead author and wrote about 80 % of the paper, based on collective experiences by the authors. I provided the theoretical lens of objectivity from philosophy.
G	Developed a conceptual model of influences on security priority based on a longitudinal study in one company (C1). Provided recommendations for what security experts could consider doing and what they should beware of in order to improve security prioritisation (C2). Provided a rich description of the experiences and the context (C4).	I was the lead author and wrote close to 100% of the manuscript. I did all the data collection and analysis.
H	Presented experiences with meetings that are in line with the idea of the Security Intention Meeting Series, identifying implications for practice (C3) and for research (C4).	I was the lead author and wrote close to 100% of the manuscript. I did all the data collection and analysis.

## DISCUSSION

The main goal of this thesis was to understand the following: *How can regular security prioritisation be integrated into agile software development so that software products end up with a level of security that is “good enough”?* This led to the identification of a design goal and knowledge questions. In Chapter 4 we described the main contributions of this thesis in relation to the knowledge questions. In this section we discuss the contribution towards the overall goal of the thesis and the design goal.

### 5.1 Research objective revisited

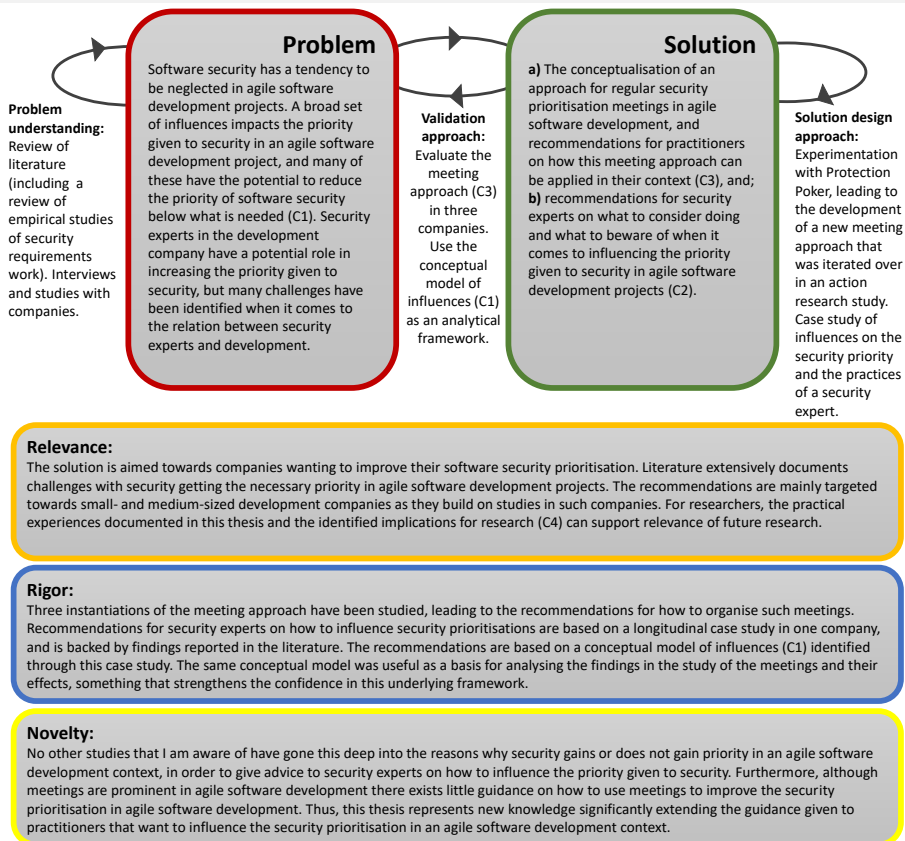
This thesis aims to tackle the design problem to *improve* software security prioritisation by developing process support for making and following up on software security priorities and decisions *that satisfies* the needs of agile development projects *in order to* support projects in achieving “good enough” security. The term *process support* can mean many things. In this thesis this has taken the form of:

- Identified and evaluated strategies that security experts can take in influencing the security priority of agile software development projects (C2), and
- A new and evaluated meeting approach for continuous software security in agile software development (C3).

Figure 5.1 visualises the design science contribution of this thesis, using a template for a visual abstract for design science [80, 81]. The visual abstract starts with stating a *technological rule* that is aimed at capturing the main theoretical contribution of the work using one phrase. Then

the visual abstract provides an overview of the *research scope and process* through describing the problem instance, the solution, and the approach taken to understand the problem and design and evaluate a solution. This is followed by a *design knowledge assessment* using the criteria of relevance, rigor, and novelty. In the following we discuss in more detail the relevance, rigor, and novelty of this work as outlined in the visual abstract.

**To improve prioritisation of security in agile software development apply regular security prioritisation meetings and empower security experts with knowledge on how to influence the security priority.**



**Figure 5.1:** Overview of the contribution using the visual abstract template for design science research [80, 81]

### 5.1.1 **Relevance**

*Relevance* is concerned with the relevance of the technological rule for a class of problems and solutions, and for stakeholders [80, 81]. This thesis addresses the problem of security prioritisation in agile software development, a problem that is extensively documented in literature. To illustrate, several systematic literature reviews highlight the problem of neglect of software security and other qualities in agile software development [2–5]. On the other hand, recent empirical studies have challenged this former evidence of quality requirements neglect. In their study, Jarzkebovic and Weichbroth [82] found that quality aspects, and in particular security, were considered important by all interviewees and that, contrary to expectations, quality requirements were not neglected. Though requiring effort, working with quality requirements were not considered problematic and interviewees talked about their attempts to identify non-functional requirements in the early phases of their projects. Similarly, Karhapää et al. [50] found less challenges than they expected in their interview-based multiple case study, and concluded that this may imply that companies doing agile software development have become more quality-aware and have found ways to handle quality requirements. Still, the studies performed in this thesis (e.g., Paper G) points to a risk of neglect despite increasing awareness of the importance of software security. Thus, there is reason for assuming that supporting security prioritisation is important for many companies now and moving forward.

The solution, in this case the recommendations for conducting meetings (C3) and the direction offered to security experts (C2), are based on case studies and action research studies in small- and medium-sized companies. These types of studies offer the possibility for analytical generalisation [75] where one may make a theoretical claim on how the findings can be applied to situations beyond the studied case, e.g., based on similarity of concepts or principles [75]. Although I do not claim that the recommendations made would be relevant for all agile software development projects, I find it likely that they can be useful for small- and medium-sized development companies that have some security expert role and are in need of improving their software security maturity. The solutions proposed are relatively simple to apply and can be adapted to the needs of the company. Such adaption is even recommended, as emerging processes was found to be more easily adopted than highly prescriptive approaches (Paper H). The solution offered is mainly targeted towards practitioners. Still, all contributions (C1-4) expand current knowledge and can help enhance the relevance of future research (C1, C4).

### 5.1.2 **Rigor**

*Rigor* is concerned with the maturity of the technological rule. This includes whether it is based on a valid understanding of the problem, whether the intervention is a valid solution to the

problem, and the validation of the design choices [80, 81]. These three aspects of rigor are discussed below. Rigor is also related to threats to validity, that are discussed in Subsection 5.2.

Software security is a complex topic in itself, with many risk factors and many potential practices to adopt. Software development can be complex in terms of the software that is developed and the situation in which the development occurs. The problem of security prioritisation in agile software development is also complex, as demonstrated by the large number of influences on security prioritisation identified in Paper G. Previous studies within agile development have found that there can be considerable variations among companies in their experiences with managing quality requirements [50, 51]. Consequently, this thesis puts a strong emphasis on understanding the problem context, both broadly (Study S1 and S3) and in specific instances (Study S4). The context has been considered in the design and analysis of the studies, and the thesis contributes to an improved understanding of what aspects of a project and its context influences the security prioritisation (C1).

The solution proposed in form of recommendations for meetings (C3) and for security experts (C2) is based on studies in companies. The grounding of this solution in empirical studies of real-life practices gives confidence in its relevance as a valid solution to the problem, or at least a step towards a valid solution. The resulting recommendations have however not been validated in other companies. Thus, it is not possible to conclude whether the developed solution is valid in a broader set of contexts. Still, being a valid solution for a given company does not mean that it is a valid solution to the overall objective that this thesis set out to contribute to. The technological rule depicted in the visual abstract in Figure 5.1 can be directly mapped to the overall objective of integrating regular security prioritisation into agile software development projects. However, the objective was also to do this so that the level of security in the product is good enough. A basic assumption of this thesis is that to reach good enough security as a strategy - and not by chance - security prioritisation is a prerequisite. However, as good enough security is hard to measure (Paper F), it is challenging to provide any hard data on whether the treatment developed in this thesis (recommendations for meetings (C3) and for security experts (C2)) leads agile software development projects towards good enough security. This thesis does not make any attempts to verify the assumption that strengthening security prioritisation is the way to go to achieve good enough security. Paper F that explored the concept of good enough security pointed to the following practices as important supporters of good enough security: including a variety of perspectives, building interactional expertise, and supporting confirmability. The meeting concept proposed in Paper E supports all these practices, indicating a potential for this meeting type to support good enough security. However, in practice, not all meeting instantiations studied (Paper H) were successful in including such a broad variety of roles. Despite an emphasis in the studies on understanding effects and adoption of the techniques

studied, it is hard to make conclusions on these aspects. This is however not a challenge only in this work but has been identified as a challenge also with previous empirical studies within this area (Paper D).

The design choices to develop recommendations for security meetings and security expert interventions were grounded in literature and in experiences made in the early phases of the work on this thesis. Regarding meetings, the meeting concept of Protection Poker [64, 65] was early on identified as a promising solution (Paper A and B), and was studied further in this thesis (Paper C). Meetings were already important in agile software development and prominent in agile methods such as Scrum [1, 56]. Regarding security experts, this role had been identified as having a potential to influence software security prioritisation [12, 49, 72]. At the same time, it had been identified that security experts often would have challenges in their interaction with software development projects due to a divide between security and development [10, 11, 76]. These findings from literature were confirmed and extended upon in this thesis. Experiences with Protection Poker led to a new meeting concept (Paper E), that then was evaluated and resulted in a set of recommendations for practitioners (Paper H). Challenges identified in the relation between security experts and development (Paper B) were further explored in a study of a security initiative led by a security expert (Paper G), in order to get to recommendations for experts wanting to influence the security prioritisation of agile software development projects. There are however other possible interventions that could have been studied and developed. This could be in form of other types of interventions (e.g., security testing), different meeting concepts, or interventions by other stakeholders than security experts.

The solution (artefact) designed ended up being less concrete than envisioned at the beginning of the work on this thesis. This happened as a response to findings - both in literature and in the studies - that emerging practices were more promising than prescribed approaches. Thus, the thesis does not propose a prescriptive practice that companies should apply but rather offer guidance and new knowledge that can support companies in building and improving upon their own practices related to security prioritisation.

### **5.1.3 Novelty**

*Novelty* is concerned with positioning the technological rule related to previous knowledge. Novelty can take the form of improved problem insights, novel solutions, or mapping of existing solutions to a known problem [80]. The knowledge provided by this thesis on influences on security prioritisation (C1) represents improved problem insight into the challenge of getting software security prioritised in practice in self-managed agile projects. Furthermore, the rich descriptions provided of the practices studied in Study S4 and S5, contribute with increased understanding of the practical challenges of software security prioritisations, as well as the



practicalities experienced by a security expert when applying a security prioritisation meeting approach (C4). The solution proposed, in form of recommendations for meetings (C2) and for security experts (C3), is novel in that I am not aware of the presence of such recommendations in existing literature. Furthermore, limited existing studies exist that can provide the basis for such advice. However, the basic idea of running meetings and involving security experts are not novel, but rather well-known practices that are applied by many companies to some extent without considering this to be a specific technique or approach. This way, the thesis both provides a mapping of well-known practices to a known problem and offers recommendations not previously available.

## 5.2 Threats to validity

Limitations to the design science contribution of this thesis were to some extent discussed above, related to the concept of rigor (Subsection 5.1.2). In the following, this discussion on limitations is continued by considering threats to validity of the overall research approach of this thesis. More discussions of limitations and threats to validity of the individual studies are provided in each paper. Validity in design science can be discussed based on criteria relevant for the empirical methods used [83]. Thus, the following discussion on threats to validity use validity criteria that are relevant for case study research in software engineering [84], adding the criteria of instantiation validity suggested for design science [85]. Table 5.1 provides a high-level overview of the discussed validity threats, and shows which of the main contributions they are most related to.

**Table 5.1:** Overview of the discussed validity threats and their relevance to the main contributions (a ‘V’ indicating relevance)

<b>Threat to validity</b>	<b>C1</b>	<b>C2</b>	<b>C3</b>	<b>C4</b>
<b><i>Construct validity and instantiation validity:</i></b>				
Broad and unclear concept <i>security prioritisation</i>	V	V	V	-
Broad and unclear concept <i>good enough security</i>	V	V	V	-
Many potential instantiations (of context and approach)	-	V	V	-
<b><i>Internal validity:</i></b>				
Measure effects on security priority and good enough security	V	V	V	V
Missing perspectives	V	V	V	V
<b><i>External validity:</i></b>				
Analytic generalisation from one or a few cases	V	V	V	V
<b><i>Reliability:</i></b>				
Dependence on the researcher	V	V	V	V

### 5.2.1 **Construct validity and instantiation validity**

Construct validity can be defined as “to what extent the operational measures that are studied really represent what the researcher has in mind and what is investigated according to the research questions” [84]. For design science, a related validity category of instantiation validity has been suggested, that refer to “the validity of IT artifacts as instantiations of theoretical constructs” [85].

A central construct related to this thesis is that of security prioritisation, a rather broad concept encompassing many potential activities and aspects. Thus, a large part of the work of this thesis has been to increase understanding of the concept of security prioritisation, including what influences the security prioritisation (C1). As security prioritisation in agile software development was not well understood in the literature, and because of the broadness of the concept, this thesis is deliberately broad in its approach. This is reflected in the initial phase of the thesis in Study S1 and S3, and in the approach to identifying influences on security prioritisation in Study S4. Such a broad concept is however difficult to define and measure. The same goes for another central construct in this thesis, that of good enough security. As discussed in Paper F and above (Subsection 5.1.2), this construct is hard to define and hard to point to and measure. We approached this concept through theory on objectivity together with practical experiences from research with companies, building a stronger understanding of the concept. However, as good enough security is a fleeting concept that may change throughout the lifecycle of a software product, it is challenging to make any clear statement that the concept of good enough security has been properly grasped or even studied.

This thesis has studied several instantiations of the problem domain and of treatments. The treatments studied (the Protection Poker, the Security Intention Meeting Series, and security expert initiatives) are all examples of possible interventions that could have been studied, and the instantiations of these treatments represent one possible way of using them in the studied context. It is quite possible that if the instantiation had happened differently this would have somewhat altered the findings. To better understand the impact of the instantiation in a context, the studies in the companies (Study S4-6) pay strong attention to the context, both in data collection and analysis. Still, there are many more contexts that could have been investigated and many other treatments and instantiations that could have been studied. The findings even emphasise that emerging techniques that are adapted to the needs of the companies should be pursued, emphasising the importance of understanding a variety of instantiations. Runeson et al. point out that to “validate a technological rule it must be instantiated, preferably in multiple cases of problem-solution pairs that instantiate the rule where each case adds to the validity strength of the rule” [83]. A strength of this work is that it studies several such problem-solution

pairs. Still, this work encourages more research into related techniques in other contexts to better understand which approaches work where and why.

### 5.2.2 Internal validity

Internal validity is concerned with causal relations and the risk that the effects observed may have been caused by factors not considered in the study [84]. This thesis grapples with understanding the effects of various interventions, such as Protection Poker (Study S2), the Security Intention Meeting Series (Study S4-S6), or other security expert initiatives (S4). For information on how the individual studies deal with concerns related to internal validity, see the papers. There is however a general challenge present in all the studies related to understanding and measuring effects. As also discussed above, both the concepts of good enough security and of security prioritisation are broad and hard to measure, and studying effects related to these concepts is challenging. This, however, does not imply that it should not be attempted. Although a concept is difficult to make clear and definite, this does not mean that it should not be scientifically studied but rather that this should be considered in study design and analysis [86].

To study effects of security initiatives and understand how these effects come about I opted for in-depth studies of selected cases, providing rich descriptions. However, no attempts were made to measure the effect of initiatives in terms of improved software security, e.g., through security testing. Thus, the evidence that any of the studied treatments gave better software security in the code is weak or non-existent. However, even if I had attempted to measure such effects through security testing, it would be very difficult to know whether any change in security level identified this way (between projects or over time) would be because of the treatment, and not because of other aspects of the context or challenges with the testing approach. Thus, the value of such testing would be limited. Thus, this work relies on informants reporting on what they experience as effects. Furthermore, the triangulation involved in data collection (observations, interviews, and documentation) strengthens the internal validity.

Although the emphasis has been on in-depth studies of cases, there are aspects of the cases that could have been studied more thoroughly. Case studies commonly cope with situations where there are many variables and where the boundary between the phenomenon and the context is blurred [75]. In this thesis, the emphasis on the security expert role has come at the cost of understanding perspectives of, e.g., developers. Thus, there may be aspects of the studied cases that are not properly grasped and perspectives that are not fully accounted for.

### 5.2.3 External validity

External validity “is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case” [84]. In the case of this thesis, what is aimed for is analytical generalisation where we make projections about the transferability of findings from individual cases. This thesis has studied several problem-solution pairs. It uses differences and similarities between these cases to make a theoretical contribution that is assumed to be relevant also for related problem-solution pairs. To support readers in evaluating whether the findings from a study is relevant for the context they have in mind, I have provided a rich description of the cases studied. This however had to be balanced with the wish for anonymity of the studied companies.

### 5.2.4 Reliability

Reliability “is concerned with to what extent the data and the analysis are dependent on the specific researchers” [84]. Although most of the work on this thesis has been done by one individual, all studies have involved other researchers. This was important to increase reliability of findings, by having more people involved in studying most of the cases and having discussions on findings from the analysis.

In the work on this thesis, care has been taken in research design to utilise my strengths to advance the research, while limiting negative effects of areas where my skills are less developed. To illustrate, although the emphasis on the security expert is grounded in a need for more research on this role, it is also based on the fact that my main background is within information security. This background made me better positioned to relate to and understand the role of the security expert than that of a developer, tester, or product owner. Thus, my ability to perform good quality research was assumed to be better when taking the perspective of the security expert. Note, however, that according to Collins [87], it is not necessary with *contributory expertise* (in this case, the skills to contribute to development) to be able to understand and study a phenomenon (in this case, software development). Rather, there is a need for *interactional expertise* (in this case, the skills to engage developers, product owners, etc., in talk).

## 5.3 Recommendations for future work

The papers provide recommendations for future work based on the individual studies. In addition to what is stated in the papers, I would however like to point towards a few more general avenues for further research based on the overall contribution of this thesis.

This thesis has considered security prioritisation in agile software development, but security is only one of several quality aspects that can be relevant for a software development project. Literature shows that quality aspects (including security) share many challenges in relation to agile software development, e.g., that they are “fuzzy” [50], that there can be a lack of recognition by stakeholders [4, 5], and that they can be neglected [4, 5, 9]. Thus, there is a potential to join forces, and strengthen the quality perspective more generally, not only considering security. This also has the benefit of avoiding a potential pitfall, where there is the need for a separate role or meeting or approach for every quality aspect, something that will end up in a challenging overall process even if all the individual approaches are easy to apply [9]. Furthermore, there is the issue of different quality aspects interacting and having dependencies [88]. Thus, an interesting avenue of further research is to understand better how agile development projects can work with different qualities in a way that is supportive of each other. Two of the secondary papers (Paper M and N) represent a step in this direction.

This thesis set out to develop process support for making and following up on software security priorities and decisions in an agile software development project. The end-result goes in this direction, providing a meeting approach and recommendations for security experts. However, the research also points to the benefit of emerging practices, as results indicate that these are more easily adopted longer term than practices that are prescribed. This illustrates a conflict that is present in the thesis, and that is described in Paper G as a choice between an emerging vs. a prescribed approach to software security. Others have discussed similar challenges using the term ambidextrous security [89]. There is a need for some top-down approach to security, as those responsible for security in an organisation need some assurance that security is adequately taken care of. Still, there are benefits of giving development teams freedom to work with security in a way that matches the needs of the team and of the software they are developing. A related challenge is that of combining emerging software security practices and self-management principles with maturity models and the need to demonstrate to customers and users that the company can be trusted with security. This challenge represents an interesting topic for further inquiry, considering the needs of various types of companies.

## CONCLUSION

The objective of this thesis was to find how regular security prioritisation can be integrated into agile software development so that software products end up with a level of security that is “good enough”. To achieve this objective, the thesis consists of studies that identify what influences the security prioritisation throughout and agile software development projects, and how security roles and activities can support an agile software development project in reaching a “good enough” prioritisation of security. An underlying assumption is that good security prioritisation is necessary for achieving adequate security within acceptable costs.

Security prioritisation is a practical problem in need of practical solutions, but there is also a need for knowledge that can contribute to a better understanding of the mechanisms behind security gaining or losing priority in a software development project. Through a design science approach, this thesis has explored security prioritisation in agile software development to get to both knowledge and practical advice. This has resulted in four main contributions:

- A conceptual model of the influences on security priority in agile software development
- Identified and evaluated strategies that security experts can take in influencing the security priority of agile software development projects
- A new and evaluated meeting approach for continuous software security in agile software development
- Rich descriptions of practical experiences with improving software security prioritisation, bridging the gap between science and practice

These represent contributions to practice and to research.

Knowledge on the influences on security prioritisation is structured into an influence model consisting of the following influence areas: driving force, visibility, motivation, room to manoeuvre, and process match. For practitioners, this influence model can be used as a basis for security experts to understand the current state of support for security being prioritised in projects, and improve upon this support. Strategies for security experts take the form of advice related to the influence model on what to consider and beware of when it comes to the influence areas of the model. Furthermore, this thesis proposes strategies for moving towards good enough security based on theory on objectivity. Promising strategies are including a variety of perspectives, building interactional expertise, and supporting confirmability. This further motivates the design of the meeting concept, the Security Intention Meeting Series, as a potential approach to achieving this. Based on studying instantiations of this meeting concept in three companies, this thesis is able to provide recommendations to companies for arranging such meetings in a manner that increases the effect of the meetings on security prioritisation and supports longer-term adoption.

For researchers, this thesis provides a model of security influences that can be used for designing further research studies in this area. The relevance of this influence model is strengthened by its strong support in literature, and represents new knowledge on security prioritisation within an agile software development context. The knowledge developed through studying the practices of security experts and the instantiations of security meetings are important for research to better understand the practical challenges and experiences faced by practitioners. Thus, a main emphasis has been made on providing rich descriptions from the studies in the companies. Literature has identified a need for more empirical studies in this area and has identified a gap between software security practices and needs in many projects today. Improved knowledge of the practical experiences of companies can help researchers improve upon existing support provided to agile development projects in terms of available practices and recommendations.

## REFERENCES

- [1] S. Türpe and A. Poller, ‘Managing Security Work in Scrum: Tensions and Challenges,’ in *Proceedings of the International Workshop on Secure Software Engineering in DevOps and Agile Development (SecSE 2017)*, 2017, pp. 34–49. [Online]. Available: <http://ceur-ws.org/Vol-1977/paper4.pdf>.
- [2] I. Inayat, S. S. Salim, S. Marczak, M. Daneva and S. Shamshirband, ‘A systematic literature review on agile requirements engineering practices and challenges,’ *Computers in Human Behavior*, vol. 51, pp. 915–929, 2015. doi: [10.1016/j.chb.2014.10.046](https://doi.org/10.1016/j.chb.2014.10.046).
- [3] H. Oueslati, M. M. Rahman and L. b. Othmane, ‘Literature Review of the Challenges of Developing Secure Software Using the Agile Approach,’ in *2015 10th International Conference on Availability, Reliability and Security*, 2015, pp. 540–547. doi: [10.1109/ARES.2015.69](https://doi.org/10.1109/ARES.2015.69).
- [4] W. Behutiye, P. Karhapää, L. López, X. Burgués, S. Martínez-Fernández, A. M. Vollmer, P. Rodríguez, X. Franch and M. Oivo, ‘Management of quality requirements in agile and rapid software development: A systematic mapping study,’ *Information and Software Technology*, vol. 123, 2020. doi: [10.1016/j.infsof.2019.106225](https://doi.org/10.1016/j.infsof.2019.106225).
- [5] A. Jarzębowicz and P. Weichbroth, ‘A systematic literature review on implementing non-functional requirements in agile software development: Issues and facilitating practices,’ in *International Conference on Lean and Agile Software Development*, Springer, 2021, pp. 91–110. doi: [10.1007/978-3-030-67084-9\\_6](https://doi.org/10.1007/978-3-030-67084-9_6).
- [6] L. Williams, G. McGraw and S. Miguez, ‘Engineering Security Vulnerability Prevention, Detection, and Response,’ *IEEE Software*, vol. 35, no. 5, pp. 76–80, 2018. doi: [10.1109/MS.2018.290110854](https://doi.org/10.1109/MS.2018.290110854).
- [7] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries *et al.*, ‘Manifesto for agile software development,’ 2001. [Online]. Available: <https://agilemanifesto.org/>.



- [8] R. Khaim, S. Naz, F. Abbas, N. Iqbal, M. Hamayun and R. Pakistan, 'A review of security integration technique in agile software development,' *International Journal of Software Engineering & Applications*, vol. 7, no. 3, pp. 49–68, 2016. doi: 10.5121/ijsea.2016.7304.
- [9] W. Alsaqaf, M. Daneva and R. Wieringa, 'Quality requirements in large-scale distributed agile projects – a systematic literature review,' in *Requirements Engineering: Foundation for Software Quality*, P. Grünbacher and A. Perini, Eds., Springer International Publishing, 2017, pp. 219–234. doi: 10.1007/978-3-319-54045-0\_17.
- [10] D. Ashenden and D. Lawrence, 'Security dialogues: Building better relationships between security and business,' *IEEE Security & Privacy*, vol. 14, no. 3, pp. 82–87, 2016. doi: 10.1109/MSP.2016.57.
- [11] T. W. Thomas, M. Tabassum, B. Chu and H. Lipford, 'Security during application development: An application security expert perspective,' in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12. doi: 10.1145/3173574.3173836.
- [12] S. Xiao, J. Witschey and E. Murphy-Hill, 'Social influences on secure development tool adoption: Why security tools spread,' in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, 2014, pp. 1095–1106. doi: 10.1145/2531602.2531722.
- [13] N. H. Chowdhury, M. T. Adam and G. Skinner, 'The impact of time pressure on cybersecurity behaviour: A systematic literature review,' *Behaviour & Information Technology*, vol. 38, no. 12, pp. 1290–1308, 2019. doi: 10.1080/0144929X.2019.15837690.
- [14] R. J. Wieringa, *Design science methodology for information systems and software engineering*. Springer, 2014.
- [15] R. A. Caralli, J. F. Stevens, L. R. Young and W. R. Wilson, 'Introducing octave allegro: Improving the information security risk assessment process,' Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, Tech. Rep., 2007. [Online]. Available: [https://resources.sei.cmu.edu/asset\\_files/TechnicalReport/2007\\_005\\_001\\_14885.pdf](https://resources.sei.cmu.edu/asset_files/TechnicalReport/2007_005_001_14885.pdf).
- [16] 'ISO/IEC 27005:2011 Information technology - Security techniques - Information security risk management,' International Organization for Standardization, Tech. Rep., 2011.
- [17] R. S. Ross, L. A. Johnson *et al.*, 'Guide for Applying the Risk Management Framework to Federal Information Systems: A Security Life Cycle Approach,' 2010.

- [18] M. Howard and S. Lipner, *The Security Development Lifecycle*. Microsoft Press, 2006.
- [19] G. McGraw, *Software Security: Building Security In*. Addison-Wesley, 2006.
- [20] C. Weir, I. Becker, J. Noble, L. Blair, M. A. Sasse and A. Rashid, 'Interventions for long-term software security: Creating a lightweight program of assurance techniques for developers,' *Software: Practice and Experience*, vol. 50, no. 3, pp. 275–298, 2020. doi: [10.1002/spe.2774](https://doi.org/10.1002/spe.2774).
- [21] C. Weir, I. Becker and L. Blair, 'A Passion for Security: Intervening to Help Software Developers,' in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2021, pp. 21–30. doi: [10.1109/ICSE-SEIP52600.2021.00011](https://doi.org/10.1109/ICSE-SEIP52600.2021.00011).
- [22] S. Miguez, E. Erlikhman, J. Ewers and K. Nassery, 'BSIMM 12 2021 Foundations Report,' Tech. Rep., 2021. [Online]. Available: <https://www.bsimm.com/content/dam/bsimm/reports/bsimm12-foundations.pdf>.
- [23] E. Terpstra, M. Daneva and C. Wang, 'Agile Practitioners' Understanding of Security Requirements: Insights from a Grounded Theory Analysis,' in *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, 2017, pp. 439–442. doi: [10.1109/REW.2017.54](https://doi.org/10.1109/REW.2017.54).
- [24] D. Bishop and P. Rowland, 'Agile and secure software development: An unfinished story,' *Issues in Information Systems*, vol. 20, 1 2019.
- [25] L. R. Saldanha and A. Zorzo, 'Security requirements in agile software development: A systematic mapping study,' Pontifical Catholic University of Rio Grande Do Sul, Tech. Rep. 087, 2019. [Online]. Available: [https://www.pucrs.br/politecnica/wp-content/uploads/sites/166/2019/07/Technical\\_Report\\_087-Leandro\\_Ripoll\\_Saldanha.pdf](https://www.pucrs.br/politecnica/wp-content/uploads/sites/166/2019/07/Technical_Report_087-Leandro_Ripoll_Saldanha.pdf).
- [26] H. Villamizar, M. Kalinowski, M. Viana and D. M. Fernández, 'A Systematic Mapping Study on Security in Agile Requirements Engineering,' in *2018 44th Euromicro conference on software engineering and advanced applications (SEAA)*, IEEE, 2018, pp. 454–461. doi: [10.1109/SEAA.2018.00080](https://doi.org/10.1109/SEAA.2018.00080).
- [27] I. A. Tøndel, M. G. Jaatun, D. S. Cruzes and N. B. Moe, 'Risk Centric Activities in Secure Software Development in Public Organisations,' *International Journal of Secure Software Engineering (IJSSE)*, vol. 8, no. 4, pp. 1–30, 2017. doi: [10.4018/IJSSE.2017100101](https://doi.org/10.4018/IJSSE.2017100101).
- [28] I. A. Tondel, M. G. Jaatun and D. S. Cruzes, 'IT Security Is From Mars, Software Security Is From Venus,' *IEEE Security & Privacy*, vol. 18, no. 4, pp. 48–54, 2020. doi: [10.1109/MSEC.2020.2969064](https://doi.org/10.1109/MSEC.2020.2969064).

- [29] I. A. Tøndel, M. G. Jaatun, D. S. Cruzes and L. Williams, ‘Collaborative security risk estimation in agile software development,’ *Information and Computer Security*, vol. 27, pp. 508–535, 4 2019. doi: [10.1108/ICS-12-2018-0138](https://doi.org/10.1108/ICS-12-2018-0138).
- [30] I. A. Tøndel and M. G. Jaatun, ‘Towards a Conceptual Framework for Security Requirements Work in Agile Software Development,’ *International Journal of Systems and Software Security and Protection (IJSSSP)*, vol. 11, no. 1, pp. 33–62, 2020. doi: [10.4018/IJSSSP.2020010103](https://doi.org/10.4018/IJSSSP.2020010103).
- [31] I. A. Tøndel, D. S. Cruzes, M. G. Jaatun and K. Rindell, ‘The Security Intention Meeting Series as a way to increase visibility of software security decisions in agile development projects,’ in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, 2019, pp. 1–8. doi: [10.1145/3339252.3340337](https://doi.org/10.1145/3339252.3340337).
- [32] I. A. Tøndel, D. S. Cruzes and M. G. Jaatun, ‘Achieving “Good Enough” Software Security: The Role of Objectivity,’ in *Proceedings of the Evaluation and Assessment in Software Engineering*, 2020, pp. 360–365. doi: [10.1145/3383219.3383267](https://doi.org/10.1145/3383219.3383267).
- [33] I. A. Tøndel, D. S. Cruzes, M. G. Jaatun and G. Sindre, ‘Influencing the security prioritisation of an agile software development project,’ *Computers & Security*, vol. 118, 2022, ISSN: 0167-4048. doi: [10.1016/j.cose.2022.102744](https://doi.org/10.1016/j.cose.2022.102744).
- [34] I. A. Tøndel and D. S. Cruzes, ‘Continuous software security through security prioritisation meetings,’ *Journal of Systems and Software*, vol. 194, 2022. doi: <https://doi.org/10.1016/j.jss.2022.111477>.
- [35] I. A. Tøndel, M. G. Jaatun, D. Cruzes and T. D. Oyetoyan, ‘Understanding Challenges to Adoption of the Protection Poker Software Security Game,’ in *Computer Security. SECPRE CyberICPS 2018*, S. K. Katsikas, F. Cuppens, N. Cuppens, C. Lambrinouidakis, A. Antón, S. Gritzalis, J. Mylopoulos and C. Kalloniatis, Eds., 2019, pp. 153–172. doi: [978-3-030-12786-2\\_100](https://doi.org/10.978-3-030-12786-2_100).
- [36] I. A. Tøndel, T. D. Oyetoyan, M. G. Jaatun and D. Cruzes, ‘Understanding Challenges to Adoption of the Microsoft Elevation of Privilege Game,’ in *Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security (HoTSoS ’18)*, 2018. doi: [10.1145/3190619.3190633](https://doi.org/10.1145/3190619.3190633).
- [37] D. S. Cruzes, M. G. Jaatun, K. Bernsmed and I. A. Tøndel, ‘Challenges and Experiences with Applying Microsoft Threat Modeling in Agile Development Projects,’ in *2018 25th Australasian Software Engineering Conference (ASWEC)*, 2018, pp. 111–120. doi: [10.1109/ASWEC.2018.00023](https://doi.org/10.1109/ASWEC.2018.00023).

- [38] I. A. Tøndel, D. S. Cruzes and M. G. Jaatun, ‘Using Situational and Narrative Analysis for Investigating the Messiness of Software Security,’ in *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM’20)*, 2020. DOI: [10.1145/3382494.3422162](https://doi.org/10.1145/3382494.3422162).
- [39] G. Brataas, I. A. Tøndel, E. Okstad, O. Løkberg, M. G. Jaatun, G. K. Hanssen and T. Myklebust, ‘The Quality Triage Method: Quickly Identifying User Stories with Quality Risks,’ in *2020 2nd International Conference on Societal Automation (SA)*, 2021, pp. 1–7. DOI: [10.1109/SA51175.2021.9507110](https://doi.org/10.1109/SA51175.2021.9507110).
- [40] I. A. Tøndel and G. Brataas, ‘SecureScale: Exploring Synergies between Security and Scalability in Software Development and Operation,’ in *Proceedings of the European Interdisciplinary Cybersecurity Conference (EICCC ’22)*, 2022, pp. 36–41. DOI: [10.1145/3528580.3528587](https://doi.org/10.1145/3528580.3528587).
- [41] S. Türpe, ‘The trouble with security requirements,’ in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 122–133. DOI: [10.1109/RE.2017.13](https://doi.org/10.1109/RE.2017.13).
- [42] L. Kocksch, M. Korn, A. Poller and S. Wagenknecht, ‘Caring for IT Security: Accountabilities, Moralities, and Oscillations in IT Security Practices,’ *Proc. ACM Hum.-Comput. Interact.*, vol. 2, 2018. DOI: [10.1145/3274361](https://doi.org/10.1145/3274361).
- [43] S. Deleersnyder, B. De Win *et al.*, ‘Presenting OWASP SAMM, OWASP SAMM v2.0 - Core Model Document,’ The Open Web Application Security Project (OWASP), Tech. Rep. [Online]. Available: <https://drive.google.com/file/d/1ZWMk4dpS3zpXjE28wi4cf5Lq6TUjeA5x/view>.
- [44] P. Chandra *et al.*, *Software assurance maturity model, A guide to building security into software development v1.0*, OWASP Project, 2008.
- [45] S. Bartsch, ‘Practitioners’ perspectives on security in agile development,’ in *2011 Sixth International Conference on Availability, Reliability and Security*, 2011, pp. 479–484. DOI: [10.1109/ARES.2011.82](https://doi.org/10.1109/ARES.2011.82).
- [46] A. Poller, L. Kocksch, S. Türpe, F. A. Epp and K. Kinder-Kurlanda, ‘Can security become a routine? a study of organizational change in an agile software development group,’ in *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW ’17)*, Association for Computing Machinery, 2017, pp. 2489–2503. DOI: [10.1145/2998181.2998191](https://doi.org/10.1145/2998181.2998191).

- [47] J. D. Blaine and J. Cleland-Huang, ‘Software Quality Requirements: How to Balance Competing Priorities,’ *IEEE Software*, vol. 25, no. 2, pp. 22–24, 2008. DOI: [10.1109/MS.2008.46](https://doi.org/10.1109/MS.2008.46).
- [48] S. L. Kanniah and M. N. Mahrin, ‘A review on factors influencing implementation of secure software development practices,’ *International Journal of Computer and Systems Engineering*, vol. 10, no. 8, pp. 3032–3039, 2016. DOI: [10.5281/zenodo.1127256](https://doi.org/10.5281/zenodo.1127256).
- [49] S. L. Kanniah and M. N. Mahrin, ‘Secure software development practice adoption model: A delphi study,’ *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 10, no. 2-8, pp. 71–75, 2018.
- [50] P. Karhapää, W. Behutiye, P. Rodriguez, M. Oivo, D. Costal, X. Franch, S. Aaramaa, M. Choraś, J. Partanen and A. Abherve, ‘Strategies to manage quality requirements in agile software development: A multiple case study,’ *Empirical Software Engineering*, vol. 26, no. 2, pp. 1–59, 2021. DOI: [10.1007/s10664-020-09903-x](https://doi.org/10.1007/s10664-020-09903-x).
- [51] T. Olsson, K. Wnuk and S. Jansen, ‘A validated model for the scoping process of quality requirements: A multi-case study,’ *Empirical Software Engineering*, vol. 26, no. 2, pp. 1–29, 2021. DOI: [10.1007/s10664-020-09896-7](https://doi.org/10.1007/s10664-020-09896-7).
- [52] C. R. Nelson, G. Taran and L. d. Lascrain Hinojosa, ‘Explicit risk management in agile processes,’ in *Agile Processes in Software Engineering and Extreme Programming*, P. Abrahamsson, R. Baskerville, K. Conboy, B. Fitzgerald, L. Morgan and X. Wang, Eds., Springer, 2008, pp. 190–201. DOI: [10.1007/978-3-540-68255-4\\_20](https://doi.org/10.1007/978-3-540-68255-4_20).
- [53] B. G. Tavares, C. E. S. da Silva and A. D. de Souza, ‘Risk management analysis in Scrum software projects,’ *International Transactions in Operational Research*, vol. 26, no. 5, pp. 1884–1905, 2019. DOI: [10.1111/itor.12401](https://doi.org/10.1111/itor.12401).
- [54] B. G. Tavares, C. E. S. da Silva and A. D. de Souza, ‘Practices to improve risk management in agile projects,’ *International Journal of Software Engineering and Knowledge Engineering*, vol. 29, no. 03, pp. 381–399, 2019. DOI: [10.1142/S0218194019500165](https://doi.org/10.1142/S0218194019500165).
- [55] J. Nyfjord and M. Kajko-Mattsson, ‘Integrating risk management with software development: State of practice,’ in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Citeseer, vol. 2008, 2008.
- [56] K. Schwaber, *Agile project management with Scrum*. Microsoft press, 2004.
- [57] O. Al-Baik and J. Miller, ‘The kanban approach, between agility and leanness: A systematic review,’ *Empirical Software Engineering*, vol. 20, no. 6, pp. 1861–1897, 2015. DOI: [10.1007/s10664-014-9340-xc](https://doi.org/10.1007/s10664-014-9340-xc).

- [58] Z. Masood, R. Hoda and K. Blincoe, 'Real World Scrum A Grounded Theory of Variations in Practice,' *IEEE Transactions on Software Engineering*, vol. 48, no. 5, pp. 1579–1591, 2022. doi: [10.1109/TSE.2020.3025317](https://doi.org/10.1109/TSE.2020.3025317).
- [59] M. O. Ahmad, J. Markkula and M. Oivo, 'Kanban in software development: A systematic literature review,' in *39th Euromicro conference on software engineering and advanced applications*, IEEE, 2013, pp. 9–16. doi: [10.1109/SEAA.2013.28](https://doi.org/10.1109/SEAA.2013.28).
- [60] E. Corona and F. E. Pani, 'A review of lean-kanban approaches in the software development,' *WSEAS transactions on information science and applications*, vol. 10, no. 1, pp. 1–13, 2013.
- [61] Z. Bakalova, M. Daneva, A. Herrmann and R. Wieringa, 'Agile requirements prioritization: What happens in practice and what is described in literature,' in *International Working Conference on Requirements Engineering: Foundation for Software Quality*, Springer, 2011, pp. 181–195. doi: [978-3-642-19858-8\\_18](https://doi.org/978-3-642-19858-8_18).
- [62] D. Leffingwell, *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional, 2010.
- [63] K. Rindell, S. Hyrynsalmi and V. Leppänen, 'Busting a Myth: Review of Agile Security Engineering Methods,' in *Proceedings of the 12th International Conference on Availability, Reliability and Security (ARES '17)*, Association for Computing Machinery, 2017. doi: [10.1145/3098954.3103170](https://doi.org/10.1145/3098954.3103170).
- [64] L. Williams, M. Gegick and A. Meneely, 'Protection Poker: Structuring Software Security Risk Assessment and Knowledge Transfer,' in *Engineering Secure Software and Systems*, F. Massacci, S. T. Redwine and N. Zannone, Eds., Springer, 2009, pp. 122–134. doi: [978-3-642-00199-4\\_11](https://doi.org/978-3-642-00199-4_11).
- [65] L. Williams, A. Meneely and G. Shipley, 'Protection Poker: The New Software Security "Game",' *IEEE Security & Privacy*, vol. 8, no. 3, pp. 14–20, 2010. doi: [10.1109/MSP.2010.58](https://doi.org/10.1109/MSP.2010.58).
- [66] D. Ionita, C. van der Velden, H.-J. K. Ikkink, E. Neven, M. Daneva and M. Kuipers, 'Towards risk-driven security requirements management in agile software development,' in *Information Systems Engineering in Responsible Information Systems*, C. Cappelletto and M. Ruiz, Eds., Springer International Publishing, 2019, pp. 133–144. doi: [10.1007/978-3-030-21297-1\\_12](https://doi.org/10.1007/978-3-030-21297-1_12).

- [67] D. Baca, M. Boldt, B. Carlsson and A. Jacobsson, ‘A Novel Security-Enhanced Agile Software Development Process Applied in an Industrial Setting,’ in *2015 10th International Conference on Availability, Reliability and Security*, 2015, pp. 11–19. doi: [10.1109/ARE.S.2015.45](https://doi.org/10.1109/ARE.S.2015.45).
- [68] C. Weir, A. Rashid and J. Noble, ‘Challenging software developers: Dialectic as a foundation for security assurance techniques,’ *Journal of Cybersecurity*, vol. 6, no. 1, 2020. doi: [10.1093/cybsec/tyaa0070](https://doi.org/10.1093/cybsec/tyaa0070).
- [69] H. Palombo, A. Z. Tabari, D. Lende, J. Ligatti and X. Ou, ‘An Ethnographic Understanding of Software ({In} Security) and a {Co-Creation} Model to Improve Secure Software Development,’ in *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*, 2020, pp. 205–220.
- [70] A. Tuladhar, D. Lende, J. Ligatti and X. Ou, ‘An Analysis of the Role of Situated Learning in Starting a Security Culture in a Software Company,’ in *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*, 2021, pp. 617–632.
- [71] D. Geer, ‘Are companies actually using secure development life cycles?’ *Computer*, vol. 43, no. 6, pp. 12–16, 2010. doi: [10.1109/MC.2010.159](https://doi.org/10.1109/MC.2010.159).
- [72] C. Weir, S. Miguez and L. A. Williams, ‘Exploring the shift in security responsibility,’ *IEEE Security & Privacy*, pp. 2–11, 2022. doi: [10.1109/MSEC.2022.3150238](https://doi.org/10.1109/MSEC.2022.3150238).
- [73] A. R. Hevner, S. T. March, J. Park and S. Ram, ‘Design science in information systems research,’ *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004. doi: [10.2307/25148625](https://doi.org/10.2307/25148625).
- [74] A. Hevner and S. Chatterjee, ‘Design science research in information systems,’ in *Design Research in Information Systems: Theory and Practice*. Springer US, 2010, pp. 9–22. doi: [10.1007/978-1-4419-5653-8\\_2](https://doi.org/10.1007/978-1-4419-5653-8_2).
- [75] R. K. Yin, *Case study research and applications*, 6th ed. Sage, 2018.
- [76] K. R. van Wyk and G. McGraw, ‘Bridging the gap between software development and information security,’ *IEEE Security & Privacy*, vol. 3, no. 5, pp. 75–79, 2005. doi: [10.1109/MSP.2005.118](https://doi.org/10.1109/MSP.2005.118).
- [77] R. Sandhu, ‘Good-enough security,’ *IEEE Internet Computing*, vol. 7, no. 1, pp. 66–68, 2003. doi: [10.1109/MIC.2003.1167341](https://doi.org/10.1109/MIC.2003.1167341).
- [78] K. Beznosov, ‘Extreme security engineering: On employing XP practices to achieve ‘good enough security’ without defining it,’ in *First ACM Workshop on Business Driven Security Engineering (BizSec)*, Fairfax, VA, vol. 31, 2003.

- [79] G. Hurlburt, ““Good Enough” Security: The Best We’ll Ever Have,” *Computer*, vol. 49, no. 7, pp. 98–101, 2016. DOI: [10.1109/MC.2016.212](https://doi.org/10.1109/MC.2016.212).
- [80] M.-A. Storey, E. Engstrom, M. Höst, P. Runeson and E. Bjarnason, ‘Using a Visual Abstract as a Lens for Communicating and Promoting Design Science Research in Software Engineering,’ in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2017, pp. 181–186. DOI: [10.1109/ESEM.2017.28](https://doi.org/10.1109/ESEM.2017.28).
- [81] E. Engström, M.-A. Storey, P. Runeson, M. Höst and M. T. Baldassarre, ‘How software engineering research aligns with design science: A review,’ *Empirical Software Engineering*, vol. 25, no. 4, pp. 2630–2660, 2020. DOI: [10.1007/s10664-020-09818-7](https://doi.org/10.1007/s10664-020-09818-7).
- [82] A. Jarzębowicz and P. Weichbroth, ‘A Qualitative Study on Non-Functional Requirements in Agile Software Development,’ *IEEE Access*, vol. 9, pp. 40 458–40 475, 2021. DOI: [10.1109/ACCESS.2021.3064424](https://doi.org/10.1109/ACCESS.2021.3064424).
- [83] P. Runeson, E. Engström and M.-A. Storey, ‘The design science paradigm as a frame for empirical software engineering,’ in *Contemporary Empirical Methods in Software Engineering*, M. Felderer and G. H. Travassos, Eds. Cham: Springer International Publishing, 2020, pp. 127–147. DOI: [10.1007/978-3-030-32489-6\\_5](https://doi.org/10.1007/978-3-030-32489-6_5).
- [84] P. Runeson and M. Höst, ‘Guidelines for conducting and reporting case study research in software engineering,’ *Empirical software engineering*, vol. 14, no. 2, pp. 131–164, 2009. DOI: [10.1007/s10664-008-9102-8](https://doi.org/10.1007/s10664-008-9102-8).
- [85] R. Lukyanenko, J. Evermann and J. Parsons, ‘Instantiation validity in IS design research,’ in *Advancing the Impact of Design Science: Moving from Theory to Practice*, M. C. Tremblay, D. VanderMeer, M. Rothenberger, A. Gupta and V. Yoon, Eds., Springer, 2014, pp. 321–328. DOI: [10.1007/978-3-319-06701-8\\_22](https://doi.org/10.1007/978-3-319-06701-8_22).
- [86] J. Law, *After method: Mess in social science research*. Routledge, 2004.
- [87] H. Collins, *Forms of life: The method and meaning of sociology*. MIT Press, 2019.
- [88] W. Alsaqaf, M. Daneva and R. Wieringa, ‘Quality requirements challenges in the context of large-scale distributed agile: An empirical study,’ *Information and software technology*, vol. 110, pp. 39–55, 2019. DOI: [10.1016/j.infsof.2019.01.009](https://doi.org/10.1016/j.infsof.2019.01.009).
- [89] D. S. Cruzes and E. A. Johansen, ‘Building an ambidextrous software security initiative,’ in *Balancing Agile and Disciplined Engineering and Management Approaches for IT Services and Software Products*, IGI Global, 2021, pp. 167–188. DOI: [10.4018/978-1-7998-4165-4.ch009](https://doi.org/10.4018/978-1-7998-4165-4.ch009).





APPENDIX

A

## PRIMARY PAPERS



## **Paper A: ‘Risk Centric Activities in Secure Software Development in Public Organisations’**

A written permission to include this material in its published form [27] has been obtained from IGI Global.

**A**

# Risk Centric Activities in Secure Software Development in Public Organisations

Inger Anne Tøndel, Department of Computer Science, Norwegian University of Science and Technology (NTNU), Trondheim, Norway & SINTEF Digital, Trondheim, Norway

Martin Gilje Jaatun, SINTEF Digital, Trondheim, Norway

Daniela Soares Cruzes, SINTEF Digital, Trondheim, Norway

Nils Brede Moe, SINTEF Digital, Trondheim, Norway

## ABSTRACT

When working with software security in a risk-centric way, development projects become equipped to make decisions on how much security to include and what type of security pays off. This article presents the results of a study made among 23 public organisations, mapping their risk-centric activities and practices, and challenges for implementing them. The authors found that their software security practices were not based on an assessment of software security risks, but rather driven by compliance. Additionally, their practices could in many cases be characterised as arbitrary, late and error driven, with limited follow up on any security issues throughout their software development projects. Based on the results of the study, the authors identified the need for improvements in three main areas: responsibilities and stakeholder cooperation; risk perception and competence; and, practical ways of doing risk analysis in agile projects.

## KEYWORDS

Agile Development, Empirical Study, Public Organisations, Risk Analysis, Risk Centric Activities, Risk Communication, Risk Management, Software Security

## 1. INTRODUCTION

Today, nearly all sectors of society depend on software systems to operate efficiently. As the dependency on software has grown, so have the threats towards these systems and the potential consequences of incidents. Though network security measures (such as firewalls and anti-virus software) can improve the security of the software systems, these only address the symptoms of the real problem: software that is crippled with vulnerabilities (McGraw, 2006).

Building security into the software, through adopting software security activities and measures in the development process, is a direct and effective way of dealing with cyber threats towards software systems. This, however, adds to the development time and cost, and this addition needs to be justified. Working towards 100% secure systems is not feasible, thus it is necessary to identify which part of the software is more critical regarding security and which activities will be most efficient and effective in

DOI: 10.4018/IJSSE.2017100101

Copyright © 2017, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

securing the software product. Taking a risk centric approach to software security means to identify what are the major risks of the particular software that is developed, and use this knowledge of risk to guide decisions regarding software security. This is commonly recommended by current secure Software Development Lifecycles (SDLs), frameworks and maturity models (Chandra, 2008; Howard & Lipner, 2006; McGraw, 2006; McGraw et al., 2016).

In many ways, security can be considered to be in conflict with the current trend of “continuous development” (Fitzgerald & Stol, 2017), reducing efficiency by delaying delivery of new features (at least in the shorter term, though costs may be saved through having to provide fewer fixes later). Agile software development uses an iterative approach to software construction, aimed at reducing development time, and prioritising value, while improving software quality and inherently reducing risk (Cockburn and Highsmith 2001). It is clear that people issues are the most critical in agile projects and that these must be addressed if agile is to be implemented successfully (Cockburn and Highsmith 2001). Even though agile methods claim to be risk driven (Beck, 2000; Eclipse, 2016), some authors have observed that risk management has been neglected in project management of agile projects (Hijazi et al., 2012; Ibbs & Kwak, 2000; Junior et al., 2012; Raz et al., 2002). It may be more difficult to establish a working process for software security activities in agile development compared to waterfall-based development, where you could more easily have mandatory or recommended security activities for the different software development phases (ben Othmane et al., 2014; Jaatun et al., 2015; Microsoft, 2009). Oyetyan et al. (2017) provide a brief overview of secure SDLs and conclude that traditional approaches to software security do not necessarily work well with agile development processes. Additionally, security is largely a systemic property, and with agile development it can be more of a challenge to have a complete view of the final system (ben Othmane et al., 2014). At the same time, agile development may come with some opportunities regarding security, e.g. to adapt to new security threats and to have ongoing interaction with customers about security.

Risk centric software security is very much related to the way developers address security in the projects. Still, other roles in an organisation (e.g. procurers, legal experts and information security experts) can have major influences on a development project’s approach to security and can have important parts to play when it comes to identifying and understanding risk, and in making risk-based decisions in the projects. About ten years ago, van Wyk and McGraw (2005) pointed out the important role of security experts in influencing and supporting the work on security in development projects. There has however not been much research on the interaction between security experts and development projects in agile development since then.

In this article, we address the following research question: How can current software organisations work with software security in a risk centric way? As implied by this research question, we study software security within development practices that are in major adoption today, meaning our context is agile development. However, whereas agile methods are centred on the activities of teams, we take a more holistic approach, including the perspectives of organisations and projects. To answer the research question, we make a mapping of the risk centric activities, practices and challenges for implementing them among 23 public organisations in Norway. This sector has been chosen for study for three reasons. First, this sector has experienced a strong security push from the authorities, causing them to prioritise security management in the organisations. As a consequence, the importance of having someone being responsible for security has been emphasised, something that makes this sector an interesting case to study when it comes to organisational influences on risk centric software security. Second, this sector’s access to legal experts makes them aware of legal requirements on security, something that increases the likelihood that software security is given some attention in software development. Third, we had easy access to this sector through cooperation with the Norwegian Agency for Public Management and eGovernment (Difi). The organisations studied have adopted agile practices for software development for some time. In the organisations, we have talked mainly with information security people, as these are in general given broad responsibility for all issues regarding IT security,

including software. To ensure that the development project perspective is included, two projects have been studied in more detail from the viewpoint of the software architects.

The article is structured as follows. Section 2 gives an overview of current research on risk management in agile development. Section 3 describes the research method used in the study. Section 4 presents the results of the study, whereas the implications of these results are discussed in Section 5, with an emphasis on making recommendations for research and practice. The threats to validity are also discussed. Section 6 concludes the paper.

## 2. CURRENT STATE OF THE ART ON RISK CENTRIC SOFTWARE SECURITY

In this section, we start with explaining what type of activities we would expect to see in software development if a risk management approach is taken to the software security work. Then we move on to present current experiences on how risk management fits with the agile approach to software development.

### 2.1. Software Security Risk Management

Software projects come with many uncertainties, including time-to market, stakeholder expectations and budget (Islam et al., 2014). Such uncertainties lead to project risks. Software risk management is a tool that can be used to manage and reason about these risks in a structured manner. Islam et al. points out that despite the existence of several risk management methods particularly suited for software projects, current research on software risk management shows that these are not well applied. Practitioners' concern is the tangible development cost that lead to project deliverables and thus direct benefits. The impact of applying an overall risk management method on a software project is unclear (Bannerman, 2008).

Security risks are one type of risk that software products face today. Within the area of cyber security, there exist many standards, guidelines and research papers that suggest different ways of managing risk and performing risk assessments. Some of the major ones are ISO/IEC 27005 (ISO/IEC, 2011), OCTAVE Allegro (Caralli et al., 2007), and the NIST Risk Management Framework (RMF) (NIST, 2010). ISO/IEC 27005 and OCTAVE Allegro are concerned with information security risk management and take an organisational approach. RMF specifies a process for integrating organisational risk management activities into system development. These documents claim that a systematic approach to information security risk management is necessary for ensuring that security activities are aligned with the organisational goals and objective, and that organisational needs for security are identified and addressed in an effective and timely manner.

If looking more closely at the type of activities recommended in ISO/IEC 27005, OCTAVE Allegro and RMF, one can identify common activities that one could expect to see when working risk centric in an organisational context. The most obvious activity is that of performing a risk analysis, including setting the scope of the analysis. Recommendations for how to do risk analysis constitute a large part of these documents (e.g. as in OCTAVE Allegro where steps 1-7 of 8 could be categorised as being part of the risk analysis activity). A risk analysis process naturally supports another activity found in all these three documents, namely making decisions on how to treat the risk. These two risk management activities (analysing risk and making decisions on what to do with the risk) can be identified in ISO/IEC 27005, OCTAVE Allegro and RMF by looking at the steps they recommend. By closer reading, one can however identify two other main activities: follow up of risk, and communication of risk. ISO/IEC 27005 contains process steps for "risk monitoring and review" and for "risk communication and consultation". Though not specific steps in the methodology, OCTAVE Allegro includes senior management sponsorship and training as important preparatory activities. RMF, which is concerned with integration of risk management and development, moves beyond just analysing the risk, by describing implementing, assessing and monitoring controls.



Communicating essential risk information to senior management is emphasised in RMF. To sum up, key activities of risk management are:

- **Risk Analysis (RA):** Including characterising the system and the risk appetite, identifying threats towards the system and assess the associated risk;
- **Risk Treatment Decisions (RTD):** Leading to requirements on controls and/or security activities needed;
- **Risk Treatment Follow Up (RTFU):** Intended to assess whether the treatments are implemented and work as intended, and to monitor changes in risks;
- **Risk Communication (RC):** Towards senior managers, but also to make sure information relevant for risk understanding (e.g. legal requirements, changes in threats) are shared between projects and between organisational units.

Risk analysis activities in software security are motivated by similar arguments as in ISO/IEC 27005, Octave Allegro and RMF; to more effectively and less expensively identify security vulnerabilities and risks and establish mitigations (Howard & Lipner, 2006), and to make better trade-off-decisions and prioritise development efforts based on risk (Chandra, 2008; McGraw, 2006). In addition, the awareness raising among project teams (Chandra, 2008) is considered important, especially when it comes to improved understanding of what factors may lead to negative outcomes (Chandra, 2008) and the ability to think like an attacker (McGraw et al., 2016). Threat modelling is even stated to be “*The Cornerstone of the [Security Development Lifecycle], and the threat model “the major [Security Development Lifecycle] artefact” that “must be used as a baseline for the product”* (Microsoft, 2009). As such, major software security frameworks, maturity models and secure SDLs include activities very much related to risk management. Table 1 shows how the Building Security In Maturity Model (BSIMM) (McGraw et al., 2016), the OWASP Software Assurance Maturity Model (OpenSAMM) (Deleersnyder et al., 2017), the seven touchpoints for software security (McGraw, 2004) and Microsoft’s secure SDL (Howard & Lipner, 2006) all contain activities that are related to the four key risk management activities described above.

Understanding and assessing security risk is known to be a complex challenge. A risk-based approach usually implies having an overview of the criticality of the various software assets, understanding potential threats and vulnerabilities (also from an attacker perspective) and being able to provide estimates on likelihood and consequences of the different types of incidents that can harm the software and impact the service the software delivers to its users. It is also necessary to understand how the various risks can be mitigated effectively. Risk analysis, and especially quantitative risk analysis, has been characterised by some as “*a modern fairytale*” in the domain of information security, as there is no overview of all threats and not sufficient data to estimate probability and consequences. To compound the challenge, there is typically not adequate method support (Oppliger, 2015). If this is the case for information security, it is likely also the case (and maybe even more so) for software security. There is limited empirical data available on what makes risk management difficult, both for information security and software security. A review of risk analysis methods for IT systems (Sulaman et al., 2013) identified a lack of evaluation of risk analysis methods. Despite the mantra that all security work should be risk based, a study among information security professionals (Jourdan et al., 2010) unveiled that as many as 25% stated that risk analysis was never or rarely performed for their department or organisation. A main challenge that has been identified regarding information security risk assessments is the estimation of likelihood and cost of information security risks, due in part to limited historical data available and constantly changing risk factors (Cybenko, 2006; Fenz & Ekelhart, 2010; Gerber & Von Solms, 2005; Rhee et al., 2012; Tøndel et al., 2015). Information is an intangible asset where it is “extremely difficult if not impossible to determine precise value” (Gerber & Von Solms, 2005), and many losses are never discovered and reported (Rhee et al., 2012).



**Table 1. Overview of how the key risk management activities are included in key software security frameworks, maturity models and secure SDLs**

	<b>BSIMM</b>	<b>OpenSAMM</b>	<b>Touchpoints</b>	<b>Microsoft SSDL</b>
<i>Risk analysis</i>	-Attack Models (Intelligence) -Architecture Analysis (SSDL Touchpoints)	-Threat Assessment (Construction)	-Abuse cases -Risk analysis	-Perform Security and Privacy Risk Assessments (requirements) -Perform Attack Surface Analysis/Reduction (design) -Use Threat Modeling (design)
<i>Risk treatment decisions</i>	-Standards and Requirements (Intelligence)	-Security Requirements (Construction) -Secure Architecture (Construction)		-Establish security and privacy requirements (requirements) -Establish design requirements (design)
<i>Risk treatment follow up</i>	-Strategy and Metrics (Governance) -Code Review (SSDL Touchpoints) -Security Testing (SSDL Touchpoints) -Penetration Testing (Deployment)	-Strategy & Metrics (Governance) -Design Review (Verification) -Code Review (Verification) -Security Testing (Verification)	-Risk based security tests -Static analysis (tools) -Penetration testing -External analysis	-Perform static analysis (implementation) -Perform dynamic analysis (verification) -Perform Fuzz Testing (verification) -Conduct Attack Surface Review (verification) -Conduct final security review (release)
<i>Risk communication</i>	-Training (Governance)	-Education and Guidance (Governance)		-Core security training (training)

## 2.2. Software Security Risk Management in Agile Development

Risk management can be said to be treated implicitly in agile development projects (Odzaly et al., 2017; Tavares et al., 2017). As explained by Nelson et al. (2008), one such implicit risk management technique is to prioritise tasks in the beginning of the iteration. As such, high-risk tasks can be prioritised, something that can reduce overall project risk. But as Nelson et al. point out, risk management is broader than prioritising high-risk tasks. There is a need to follow up on the risk and take additional actions if necessary.

As the guidance provided by agile methods when it comes to risk management can be said to be “very general” (Nyfjord & Kajko-Mattsson, 2008), research has aimed to tailor risk management to agile development projects, with several approaches being suggested (Odzaly et al., 2017). Few studies have however covered integration of risk management with agile software, considering the organisational level (Nyfjord & Kajko-Mattsson, 2008; Odzaly et al., 2017).

On security risk management for agile projects, the research papers are even fewer. There are few studies on practices, and few risk analysis methods specifically tailored towards agile development. One of the more relevant studies available is an action research study at Ericsson, (Baca et al., 2015) of the effects of implementing a security enhanced agile software development process (SEAP). This process included several software security activities (e.g. code review, penetration testing), however the study focuses solely on two key aspects of SEAP: Adding more security resources in the project and the development teams, and performing incremental risk analysis. The introduction of SEAP

was found to improve identification and handling of risk, and because of this, the risk-management is found to be more cost-efficient than with the approach previously used by Ericsson. This is claimed to be due to security issues now being dealt with in a more distributed fashion, and thus more issues are solved directly by the team. The details of how risk analysis and risk management was with SEAP is not available, apart from describing that frequency of risk analysis was being increased, the scope for each analysis was being reduced, and the approach was becoming more distributed.

The most notable risk analysis method available that is specifically tailored to agile development is Protection Poker (Williams et al., 2010), a game for risk assessment to be used by agile teams. Evaluations of adoption of Protection Poker in real development projects is however sparse. There is one study available where Protection Poker was used by one team at RedHat (Williams et al., 2010), but this study focused more on awareness and knowledge raising through using the techniques, and not on adoption. And despite positive evaluation results, the team studied stopped using Protection Poker sometime after the study was ended.

We are not aware of any studies or specific methods that aim to understand or solve software security risk management in agile development, taking an organisational approach. Such research is however much needed, as effects of security interventions in agile development have been found to be dependent on organisational factors (Poller et al., 2017).

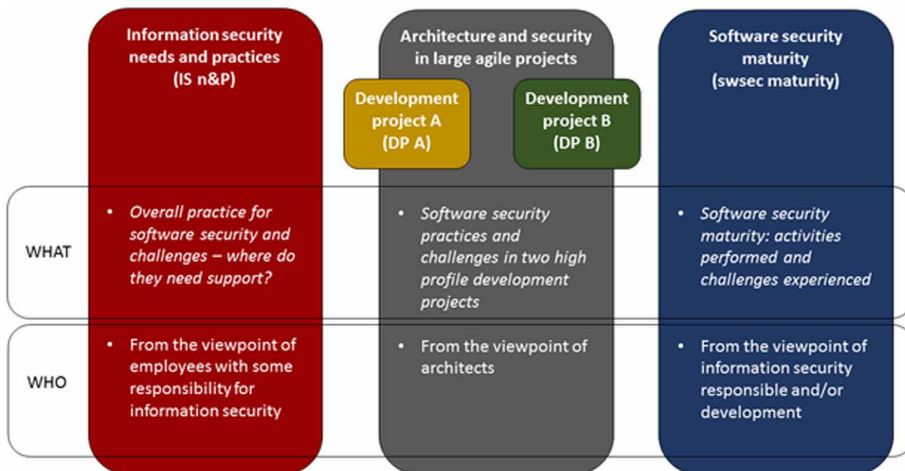
### 3. RESEARCH METHOD

This section gives an overview of the research method used for the study. It starts with describing the study goal and study design, explaining how the study consists of three separate sub-studies. Then it moves on to explaining the sub-studies in more detail, before describing the approach to analysis.

#### 3.1. Research Questions and Study Design

This study is a combination of three individual studies (sub-studies) that have been performed over the span of two years and that address a common theme. Overall, the study is motivated by the vital role a risk centric approach is considered to have in the literature when it comes to achieving cost effective software security and it tackles the research question “How can current software organisations work with software security in a risk centric way?” Figure 1 shows how the three individual sub-studies

Figure 1. Overview of the individual sub-studies





together address this research question from three different angles. The first sub-study studied software security as part of the overall information security management practices in the organisations, from the viewpoint of people working on information security. By having this as the first sub-study we were able to get a high-level view of the overall challenges and status on an organisational level. However, this sub-study deliberately ignored the developer viewpoint, and thus missed a central perspective on the topic studied. Thus, in the second sub-study we studied two high-profile public development projects in more detail, aiming to get an understanding of how security was handled in the projects. Together, sub-study one and sub-study two gave an overview of practices and challenges related to software security from both the viewpoint of security people and software architects in the development projects but lacked a more structured overview of the software security activities adopted in the organisations. As such, sub-study three aimed to get such an overview by mapping their software security activities to those described in the BSIMM framework (McGraw et al., 2016). The three sub-studies together made us able to identify and map risk centric activities and practices in the organisations and understand challenges of implementing a risk centric approach to software security.

As explained in the introduction, the decision to study development in the Norwegian public sector was made based on three factors: a security push in this sector had forced them to prioritise security management in the organisations, legal expertise in the organisations made them aware of any compliance requirements on security, and we had easy access to study participants. The Agency for Public Management and eGovernment (Difi) was our partner in two of the sub-studies (sub-study 1 and 3), contributing with financing as well as helping in participant recruitment. This made it possible to get access to a high number of organisations within a sector where security was receiving growing attention.

Table 2 gives an overview of key facts about the sub-studies. As can be seen the full study has been performed over a period of two years. As some organisations participated in more than one sub-study, the total number of public organisations studied is 23, in addition to two software companies (consultants/contractors) that had a central role in public software development projects.

**Table 2. Key facts about the sub-studies**

	<b>Information Security Needs and Practices (IS n&amp;p)</b>	<b>Architecture and Security in Large Agile Projects (DPA, DPB)</b>	<b>Software Security Maturity (swsec Maturity)</b>
<i>When</i>	2013	2013	2015
<i>Public organisations</i>	13	1	20
<i>Software companies (consultants/contractors)</i>	-	2	For some organisations, consultant developers gave input to the questionnaire
<i>Data collection method</i>	Focus group interviews	Group interviews and documentation	Questionnaire with follow up interview
<i>Study participants</i>	Information security/network security	Architects	Some information security, some development
<i>Level of focus</i>	Organisation	Project	Activities

### **3.2. Sub-Study 1: Information Security Needs and Practices (IS n&p)**

In the first sub-study, the aim of the study was to identify public organisations' needs for support regarding information security, and the study covered several topics of which software security was one. The study was performed using the focus group technique (Stewart & Shamdasani, 2014). A focus group can be understood as a semi-structured group interview, and the technique is well suited to identify areas of improvement based on the experience of the participants. By bringing the participants together, rather than performing individual interviews, the participants relate to the opinions of the others in a conversation, something that brings out more information.

Invitations to participate in the study were sent to 26 public organisations, out of which 13 agreed to participate. The invitations to participate were sent by Difi, who also selected which organisations to invite. The main criterion used in the selection was that the participating organisations should include a mixture of organisations regarding both size and security maturity. In addition, a few organisations that were known to have received critical remarks from the Office of the Auditor General of Norway (Riksrevisjonen) regarding information security were invited to participate. The invitations requested the participation of personnel that had some degree of responsibility for information security in their organisations.

In total, three focus group interviews related to software security were performed, and each focus group interview lasted in total three hours. One group consisted of organisations that were believed to be mature in their information security work. The other two groups were more mixed in terms of participants. Few of the participants knew each other from before, and we used brainstorming techniques in the start of each focus group to build trust among the participants, as a successful focus group is dependent on participants that are willing to share experiences.

All groups followed the same process and interview guide. After a short introduction to the study, we performed a short brainstorming where participants addressed the following question: What works well and what is challenging in the work with information security in your organisation? Then we started on the group interview, covering the following topics: 1) security culture and management buy-in 2) information security management systems and risk management, and 3) software development. In the software development part, the participants were asked about when in the process (requirements, development, deployment) information security people and activities were involved, and challenges were discussed. In addition to the questions from the interviews, the later focus groups were presented with important findings from previous groups and asked to comment on those findings.

Two researchers facilitated all the focus groups. One of these was responsible for taking detailed notes from the discussion. In addition, all conversations were recorded. After each focus group, the researchers reflected about how the groups functioned; whether everybody participated in the discussion, if they agreed a lot or disagreed. Observations regarding group dynamics have been taken into account in the analysis of the results, in addition to other context information regarding background and experiences of participants and maturity of their organisations. After each focus group, a summary was made and sent to the participants for comments. The summary included a recapitulation of the most interesting points of the discussion, as well as unstructured and anonymised notes from the discussion.

### **3.3. Sub-Study 2: Architecture and Security in Large Agile Development Projects (DP A and DP B)**

In the second sub-study, one very large agile project was selected as the primary case to study. In the following we call this project "Development project A". This project ran for four years and consumed roughly 800 000 man-hours and was among the largest agile software projects in Norway at that time. In total three organisations were involved in the development, this was the public organisation itself and two contractor organisations. For more details on this case and the study design see (Dingsøy et al., 2017).

In this sub-study, we performed interviews on how architecture and security was handled in the project. We organised three group interviews, one for each organisation. The reason for having

only one organisation in each interview was that we wanted to capture potential differences between the organisations. Participants to the group interviews were recruited to the study by asking the involved organisations to invite the most relevant people for the topic. For the group interviews on software architecture and security only one of the contractors was able to participate with personnel that had worked on the Development project A, resulting in two group interviews on this topic. In total six persons from the project participated in these two interviews; three persons from the public organisation and three from one of the contractors. All these had served in software architect roles at some time during the project, ranging from team architect to chief architect. The interviews were performed by two researchers, recorded and transcribed.

The contractor that could not participate with personnel from Development project A, instead participated in the study with one experienced software architect that was responsible for another high-profile public development project. In the following we call this project “Development project B”. This agile project was considerably smaller than Development project A, with only two teams and one contractor, and one year of development. However, this project had a considerably higher focus on security issues, due to the sensitivity of the data handled.

### **3.4. Sub-Study 3: Software Security Maturity (swsec Maturity)**

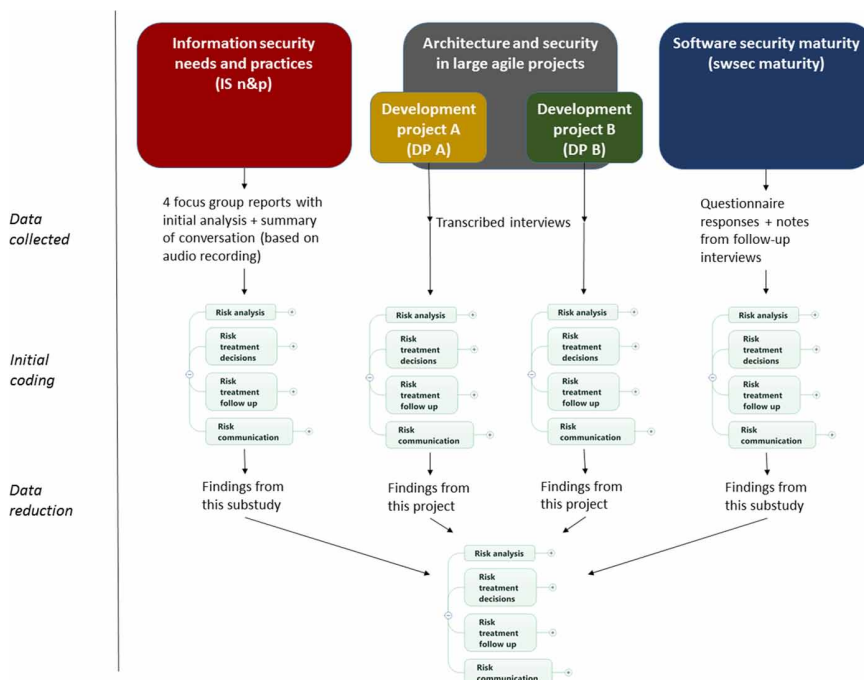
The third sub-study was performed about one and a half years after the other two sub-studies, and was aimed at measuring the software security maturity among Norwegian public organisations (Jaatun et al., 2015). The study instrument consisted of a questionnaire based on the Building Security In Maturity Model (BSIMM) as documented in the BSIMM V report (McGraw et al., 2013). The main function of BSIMM is to serve as a yardstick to determine where an organisation stands compared with other organisations. The questionnaire tells us what activities the organisation has in place and based on how well they cover the various practices, we can determine the maturity level of each organisation. The questionnaire was followed up by an interview, in order to verify the answers and clarify potential misunderstandings in filling out the questionnaire. In most cases, minor updates to the questionnaire was made based on the interviews. Each interview was performed by one researcher, they were conducted online, and were recorded. Three researchers cooperated on performing the interviews, and these researchers discussed any ambiguities in the questionnaire beforehand to ensure that their assessments of activities were as similar as possible.

The questionnaire was distributed via email to 32 Norwegian public organisations which we had reason to believe had ongoing software development activities. 20 of these organisations returned fully filled-out questionnaires. For seven of the responses, the questionnaire had been filled out in cooperation by representatives involved in software development and in general IT security work. In the other cases, the response was made either by people working on information security or on IT in general (six responses), by people working on software development (five responses), or the main responsibility of the respondent was unclear based on the job title (two responses). In most cases, at least one of the respondents had a managing role in the organisation, e.g., information security manager, IT manager, group leader or architect. The resulting questionnaire responses were analysed to find the maturity, and the results from the analysis of the questionnaire is documented elsewhere (Jaatun et al., 2015). In this paper, we have also analysed the notes from the follow-up interviews, and use these to shed more light on the questionnaire responses and the software security practices reported.

### **3.5. Analysis**

The three studies were aimed at identifying existing practices and challenges when it comes to software security in these organisations. To understand these practices and challenges, we coded the data based on the assumption that risk management is a key to making decisions regarding software security, as depicted in Figure 1. The overall process of analysis is illustrated in Figure 2. For the analysis, we used Mind Manager, with one separate mind map for each study. To ease comparison of data from the different studies, the overall structure of each mind map was the same: risk assessment,

Figure 2. Overview of process for analysing the data



risk treatment decisions, risk monitoring, risk communication. Then the data within each of these overall topics was organised into categories based on the coding. After coding and creating a mind map for each sub-study, key findings from each of the categories and for each study were identified, and these were again used to organise the data into a mind map that included the key findings and data from all the studies, and analyse these together.

## 4. RESULTS

In this section, we give an overview of practices and experiences from the organisations and projects studied when it comes to risk analysis, risk treatment decisions, risk treatment follow up and risk communication. Then we provide an overview of triggers and barriers not directly related to risk management that seem to be important for software security in the studied organisations.

### 4.1. Practice Adoption

Table 3 gives an overview of the main findings on adoption of risk centric practices in the studied organisations. The state of adoption is summarised in the findings-column with a number. The meaning of this number differs between the sub-studies. For sub-study 2 (DP A and DP B) the adoption is related to an individual project, and shows if the project adopts the practice throughout the project (2), adopts the practice to some extent (e.g. does it in an ad-hoc manner or partially) (1) or does not adopt the practice (0). For sub-study 1 (IS n&p) and sub-study 3 (swsec mat) the practice adoption relates to organisations, not projects, and the number assigned depends on the practices of the studied organisations overall. Thus, a practice is said to be adopted (2) if most of the organisations (80% or





Table 3. Overview of adoption of risk management activities

Main Findings		Activity	Practice (2 = Adopted; 1 = Sometimes/to Some Extent; 0 = Not Done; '-' = Unclear/No Data)				Notes on Adoption
			IS n&p	DP A	DP B	Swsec mat	
<i>Risk Analysis</i>	Risk analysis practices vary greatly among organisations (RA1)	Characterize system	-	1	2	1	No clear process for this. Guided by risk perception where confidentiality has the main priority (less focus on integrity and availability).
	Legal requirements are a driver for performing risk analysis (RA2)	Identify threats/ threat modeling	-	0	-	1	Some organisations are starting to do this more, but not common practice yet.
	Risk analyses are often not centred on software security issues (RA3)	Analyse risk	1	0	2	1	Often not relevant (high level, not specific on security risk). Determination of security needs is more driven by compliance than risk.
<i>Risk treatment decisions</i>	Arbitrary, late and error driven (RTD1)	Security require-ments	1	1	2	1	In many cases, security requirements are considered rather late in the process.
	No one fights for software security (RTD2)	Secure design and architecture	-	0	2	1	
<i>Risk treatment follow-up</i>	Legal requirements dictate specific security measures – creates tension (RTD3)						
	Most trust the vendors and the developers to follow up security (RTFU1)	Code review	-	0	1	1	When code review is done, its focus is not security, but other code quality aspects.
	Limited security testing or review (RTFU2)	Design review	-	0	-	1	
	Time pressure results in security requirements being postponed (or even dropped) (RTFU 3)	Security testing	1	0	2	1	Most testing is done on general functionality.
		Metrics	-	0	-	0	
<i>Risk communication</i>		Monitor changes in risk	-	0	-	1	Organisations monitor changes in network security risks. Monitoring of changes in risks related to sw development is dependent on developer interest.
	Lack of training in software security risks (RC1)	Training in sw security	0	-	0	1	
	Silo structure prevents spread of knowledge (RC2)	Sharing of risk information within organisation	1	-	-	1	Legal expertise and security expertise in the organization do not necessarily benefit development projects.
		Security people involved in sw projects	1	-	2	1	Some involvement, but in most cases this involvement is very limited in terms of effort.



more) in the sub-study have adopted the practice, and it is partially adopted (1) if more than two of the organisations have adopted the practice. Note that in sub-study 1 we did not ask the organisations about their practices individually, thus practice adoption is based on our impressions from the discussions in the group interviews. Mind maps that show the basis for the findings summarised in the table can be found in Appendix A. In Table 3 we describe the findings in more detail.

Practices and routines for *risk analysis* varied greatly among the studied organisations (RA1). The risk analysis practices were most thoroughly covered by the swsec maturity sub-study (sub-study 3) where we specifically asked each organisation about their practices. In this sub-study, risk analysis was by no means an uncommon practice related to the projects. Only one organisation was clear that they never do such an analysis related to security. However, at the same time only three of the interviewed organisations stated that they always (or most of the time) do risk assessments related to software security in all development projects. In most organisations in this sub-study, security risk analysis was done only for some of the projects, and the organisations did not seem to have any clear strategy for deciding when a risk analysis was needed - the process seemed to be ad hoc and dependent on key people (such as software architects) and their interest in and awareness about security. A few informants claimed that security is more likely to be handled when they procure systems (instead of developing the systems in-house) and for new systems (as opposed to improvement of existing systems), but this appeared to be an observation of own practices rather than an intentional strategy. The IS n&P sub-study (sub-study 1) supports the impression that software architects and legal experts are key to having security needs considered in the projects.

In the swsec maturity sub-study we found that although the organisations self-reported that they do perform risk analysis related to development, such risk analysis did not necessarily cover software security risk in the projects (RA3). Risk analysis performed related to the specific development projects could be centred on other types of project risk, ignoring security risks. Risk analysis performed related to security risks could have been performed on an organisational level but were not considered relevant on the project level. Thus, current risk analysis practices do not necessarily improve software security.

In the organisations studied, the strongest driver we found for performing risk analysis related to security is legal requirements (RA2). In the swsec maturity study (sub-study 3), three of the organisations referred to legal requirements when they talked about their risk assessment practices, and one clearly stated that an audit was the trigger for performing risk assessments. Other motivations are not expressed to the same extent by the interviewees in this sub-study. In the development projects studied (sub-study 2) this effect of legal requirements is shown in practice. Development project B is the only one of the two with clear legal requirements on security, and the only one where risk analysis was done regularly throughout the project. Although development project A was high profile, security risks were not systematically analysed in the project, and the software architects we talked to did not seem to be particularly concerned about, nor updated on, the security of the system.

Risk treatment decisions are often not made based on a process that ensures a thorough understanding of risk. In the organisations studied it seems a bit arbitrary whether or not security is considered for the projects (RTD1). Although some security issues may be considered early on, important security issues may still be left out. Security is in many cases included a bit late and inclusion is error-driven. In the development projects studied this is especially the case in development project A, where security issues that were accidentally discovered was a main reason for the security efforts that the interviewees told us about. Some “surprises” are however hard to avoid, and this was also the case in development project B (where security was considered throughout the project) where the interviewee stated that “...we took some measures towards the end, where it did just strike us ‘oh, we just have to secure this.’” In this project, however, most security measures were initiated in a more planned and proactive way. All sub-studies agree that software architects have a potentially important role when it comes to security in the projects, but this is dependent on their personal initiative and interest in security. In practice, few software architects seem to have security as a main interest, and their explicit responsibilities when it comes to security are limited. Security people are sometimes

involved, but seem to be passive, either waiting to be invited or participating in the beginning and then leaving the project to fend for itself. Some state that they trust vendors to take care of security, including identifying how much and what type of security is needed.

The responsibility for identifying and deciding on security requirements for the development projects seems fragmented (RTD2). In the swsec maturity study, the interviewee from one organisation pointed to a hired consultant as the one having an interest in software security. An interviewee from a different organisation stated that it is difficult to identify who in the organisation is responsible for software security, and the interviewee considered each developer to be responsible for their part of the code. In the IS n&p substudy, participants pointed out that security is often considered a technical thing that is assumed to be covered since technical people are involved. In the same sub-study, they however explain that they are seeing an improvement in the way security is handled. This is related to people from the business side beginning to get more concerned about security. As they provide input to requirements, they have a potential role in bringing security requirements to the projects, although their competence on security is low. In general, it seems unclear in most organisations where the responsibility of information security people ends and the responsibility of the development part of the organisation starts when it comes to software security.

Legal requirements are an important source for security requirements in the development projects in the organisations studied (RCT3). Though legislation clearly motivates security efforts, there are two main problems with this. First, the legal requirements may have unintended effects when it comes to security. The architect from development project B explained that requirements to physically delete data (not only mark it as deleted) increased the risk that data is lost. Additionally, legal requirements were resulting in more complex solutions, something that is not necessarily beneficial for security. Second, there is a need to balance security and other issues. The architect from development project B stated: “Getting the legal people on board was perceived as one of the greatest challenges...” The same architect characterised security people as extreme, wanting to remove network connections etc. Making compromises is challenging when dealing with legal requirements. However, the architect explained that they sometimes had to go to the legal experts to challenge legal requirements and postpone such requirements to later iterations.

Risk treatment follow up is not done in any structured manner in most of the studied organisations. Activities such as testing and code review are common, but they are rarely considering security aspects (RTFU2). Developers and vendors are trusted to take care of security, without this being followed up by most organisations (RTFU1). Interviewees in the swsec maturity sub-study offer statements such as (paraphrased) I am sure our vendor has routines in place and I have raised this issue with the developers, but I am not sure what developers do about it. In many of the organisations in this sub-study, external vendors do most of the development, something that is mentioned as one hindrance for involvement and follow up by security people. However, one of the more mature organisations in the swsec maturity sub-study was aware that security needs to be followed up on more closely, as expressed by one interviewee (paraphrased): Contract terms state that the vendors should follow the security rules. But it is the daily practice that matters. That is why it is essential to have a process to follow up quality requirements regarding security.

Time pressure and agile development is considered a challenge when it comes to following up on security throughout the development projects (RTFU3). In the IS n&p sub-study, participants explained that “...it was easier when projects were run using the waterfall development model, because then all requirements were there before the project started, and the vendors had to deliver everything in the requirements. Today important decisions that involve information security are made in sprint meetings...” (paraphrased from focus group discussion). Security people do not seem to be present in these meetings, and thus their influence on these decisions is very limited. The swsec maturity sub-study confirmed that existing regimes on software security do not work as well as before due to agile development, further hampering ability to follow up on risk treatment in these projects.

Risk communication is in general not addressed in a structured manner in the studied organisations. The training activities were most thoroughly identified in the swsec maturity sub-study. There it was found that none of the studied organisations have a structured approach to software security training (RC1). They may have general security training for employees, but this does not cover software security. In the swsec maturity sub-study one of the interviewees stated (paraphrased): New employees have to receive a mandatory security introduction and sign the security policy, but there is nothing about software security there. And since the developers are hired consultants none of them has to go through this process. In some organisations, employees have been sent to courses or conferences where software security has been a topic. A few are also aware of training activities at the vendors, but in most cases any software security training is ad hoc and dependent on developer interest.

As has been pointed out above, legislation is an important source for security requirements. Of all the organisations in the swsec maturity sub-study, 85% self-report that they have an overview of regulations. However, in the follow up interviews it becomes clear that although the organisation may have a legal department with a clear overview of legislative requirements, this does not necessarily benefit the development projects. In the same way, security policies may be present at an organisational level, without this affecting the development, and security expertise and follow up on new threats that is done on an organisational level does not necessarily lead to this knowledge being available to development projects (RC2). To illustrate, we paraphrase the following statements from the swsec maturity interviews: We have a forum to discuss cyber-attacks in operation, but not sure if things from this forum get to developers. I hope so. And, our organisation has many policies that ensure we are compliant, but I am not sure how much this impacts the coding.

#### **4.2. Triggers and Barriers for Software Security**

As can be observed from the results described in the above subsection, the results from this study downplay the importance of risk analysis in current software security practices in many of these organisations. It is suggested that any risk analysis performed is often not relevant, and that the motivation is mainly legal compliance and not improved security per se. Additionally, risk analysis is not put forward as an important basis for making decisions about what security measures and activities are needed in the projects. Follow-up of any such decisions is also not risk-based, and very few have any clear processes to review and test for security issues and ensure that any security requirements are adequately dealt with in the final software. As can be seen in Table 3, adoption of typical risk management activities (as described in section 2.1) is low, except in development project B. Although the software security approach of these organisations does not seem to be risk centric, they do report that they perform a number of security activities, as can be seen in Figure 3 that gives an overview of the questionnaire responses in the swsec maturity sub-study (for more details on these responses, see (Jaatun et al., 2015)).

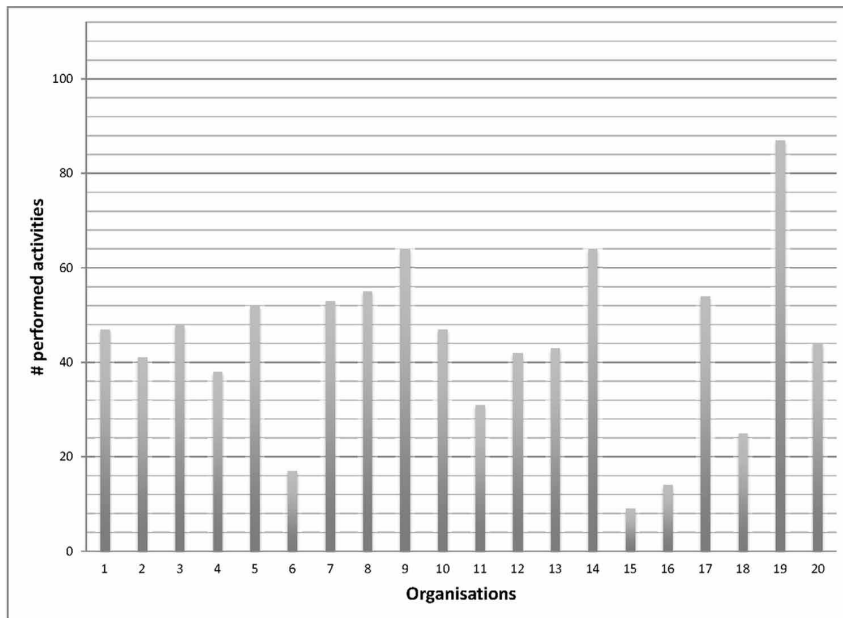
As risk management does not seem to be the most important starting point for software security in the studied organisations, we have identified from the data other activities or factors that are stated as possible triggers or barriers for software security by the interviewees (see Table 4). These have been identified by going through the coded data looking for what interviewees describe as causes for or hindrances for software security in their organisations and projects.

As described in the previous sub-section, external requirements clearly represent a trigger for performing risk analysis, and they impact the security requirements of projects as well as the effort put into security. Development project B had quite strict legal requirements related to the data the product was to handle, and this is likely the major reason why this project had a relatively strong focus on security. Additionally, the information security people we talked to in the IS n&p sub-study (sub-study 1) also saw closer interaction with the legal department as a possible way to push more security to the projects.

As also mentioned in the previous sub-section, detection of security-related errors made in code clearly triggers a burst in security attention in order to fix discovered issues. Hearing about security



Figure 3. Self-reported adoption of BSIMM activities in swsec maturity sub-study (questionnaire responses)



problems in previous projects can also increase general attention about security. However, when it comes to issues related to risk perception and attitude and competence on security, awareness seems to be low in the majority of the projects in these organisations. One of the reasons stated for this is that many of the systems they develop are meant only for internal use and will not be directly accessible from the internet. And they state that in cases where they develop open systems, these often only present data that is considered open as well, thus security is not considered important. The exception to this among the projects we have studied is development project B (sub-study 2), where the resulting project was to be accessible from the internet and contained sensitive data, and in this case, security was made a priority. In the study, we have not made any efforts to identify security risks to the software the organisations develop, in order to verify such claims. However, it is worth noting that the security experts in the IS n&p sub-study stated that awareness about the confidentiality aspects of security is much higher than awareness about integrity and availability aspects. The same information security people additionally experienced that knowledge of security was low among many of the developers and procurers, who consider security to mainly be a technical issue that will automatically be covered. Still, the overall study also gives indications that knowledge about security is improving among developers, as some of the people we talked with in the swsec maturity study (sub-study 3) perceived developers as knowledgeable about security. Furthermore, although it seems that interest in security among software architects and developers may be a bit low based on the data we have, it also shows that security can be perceived as something that makes a project interesting and challenging from a technical viewpoint.

In the organisations we have studied, the responsibility for software security seems unclear and fragmented. Few can really claim to have a software security group (SSG), i.e., a group that includes roles such as security architect or security tester. In many organisations, the responsibility for software security is considered part of the responsibility for information security, but in that case, it runs the risk of getting deprecated. Some of the informants were clear that in their organisation other security

Table 4. Triggers (↑) and barriers (↓) for software security activity in the projects

Factor	IS n&p	DP A	DP B	swsec Maturity
External requirements	↑ Legal requirements and Interaction with legal department		↑ Legal requirements	↑ Legal requirements trigger risk analysis
Errors made	↑ Security mishaps, also from previous projects	↑ Security mishaps detected	↑ Security mishaps detected	
Risk perception	↓ Systems not directly connected to the internet	↓ Systems not directly connected to the internet		↑ Project with obvious security needs
	↓	↓		↓ Systems not used by externals ↓ Open data
Responsibility	↓ Unclear responsibilities for sw security		↑ Security included as user stories ↑ Budget for security	↓ Unclear responsibilities for sw security ↓ Rely on contractors to deal with security – limited follow up
Architects	↓ Architects do not take responsibility for security	↓ Architects not that interested in security		↓ Few architects with security interest
Changes	↓ Time pressure ↓ Agile development		↓ Need for progress to fulfil contract ↓ Balancing security with other needs	↓ CISOs do not have resources to follow up on sw security
Attitude and competence	↓ Lack of security knowledge among developers ↓ Procurers lack awareness about security ↓ Security is considered mainly a technical issue	↓ Lack of awareness of security implications	↑ Security was considered part of what made the project technically interesting and challenging	↓ No/little training in sw security
Other issues				↑ New products

activities and goals had been prioritised, and consequently software security had not been given attention. Architects are seen as potential allies in a security push, but as of now they seldom have explicit responsibility for security. By and large, it is currently not possible to clearly hold anyone accountable for software security.

When changes need to be made throughout the project, security is not necessarily considered. More often than not, time pressure in the projects results in security requirements being deprecated in order to reach other project goals. This is considered a bigger problem with agile development than in more traditional waterfall-based development models.

## 5. DISCUSSION

Based on the results of these studies we have identified three main areas that is important to consider in order to improve current practice and to guide further research. These are: responsibilities and stakeholder cooperation; risk perception and competence, and; practical ways of doing risk analysis.

In the following subsections, these areas are discussed and some recommendations are made. Towards the end of the section we provide a discussion of the validity of the results

### 5.1. Responsibilities and Stakeholder Cooperation

The empirical basis for the discussion of responsibilities and stakeholder cooperation can be found in Table 5. This table shows the most relevant results from the four risk management areas identified in Section 2, including findings from Table 3 (main findings, practice adoption) and Table 4 (triggers, barriers). In this, and the following, tables, a ‘-’ in one row indicates that there is no relevant finding for this topic. In some cases, rows are merged, and in that case that means that a finding is relevant for more than one risk management area.

One major obstacle for software security in the studied organisations is the unclear responsibilities for software security. Responsibility could be put either on the security experts or on the development projects, but we recommend that responsibility is given to the projects or someone close to development. The reason for this recommendation is twofold. First, when software security is seen as part of information security, it risks being deprecated. This happened in the studied organisations. Due to a push from the government on implementing an Information Security Management System (ISMS), i.e. ISO/IEC 27001, employees having the role of Chief Information Security Officer (CISO) or similar were under pressure to improve their ISMS and relevant practices, while the pressure to improve software security were not as strong. Thus, it is not a big surprise that software security was not given priority by CISOs. As pointed out by van Wyk and McGraw (2005), the alignment of information security and development would require a large effort from information security people. In the organisations we studied, CISOs and other information security employees were already struggling with limited resources and thus unable to properly address all important security tasks. Second, the competence needed to work with information security at an organisational level does not necessarily match the competence needed to work with security in development projects. The BSIMM is clear in its recommendation that the ones responsible for software security should have a development background, stating “Starting with network security people and attempting to teach them about software, compilers, SDLCs, bug tracking, and everything else in the software universe usually fails to produce the desired results. Unfortunately, no amount of traditional security knowledge can overcome a lack of experience building software.” (McGraw et al., 2016) Training in security may however be needed for the people that is given this role.

Assigning clear responsibilities for software security does not however remove the need to improve cooperation among a variety of stakeholders related to software security in a project. As can be seen from Table 5, communication and cooperation between the participants in the development project, security people and legal experts is seen as a challenge. This is challenging in two ways: Getting these different roles to interact to have the competence of the legal experts and security experts benefit the development projects, and having these roles understand each other’s perspectives so that good trade-offs can be made. This seems to be even more challenging when development is done by vendors/contractors, limiting the interaction. It is important that organisations are aware of both the benefits and challenges of having these roles involved in development projects and make arrangements to support cooperation.

It’s been more than ten years since van Wyk and McGraw (van Wyk & McGraw, 2005) called out for aligning information security and software development. They pointed out that the disconnect between security and development led to software development without any understanding of technical security risk, and thus software with security weaknesses that should have been avoided. In their article, van Wyk and McGraw discussed the role of information security in the software security touchpoints and provided recommendations to information security people that wanted to play a part in improving software security. The role of security teams for adoption of secure development tools was further studied by Xiao et al. (2014) as part of interviews with 42 professional software developers. They found that the relationship between the security team and the developers can act

**Table 5. Empirical basis for discussion and recommendations on responsibilities and stakeholder cooperation**

Area	Main Finding	Relevant Triggers	Relevant Obstacles	Notes on Adoption
Risk analysis	-	-	-	Lack of clear processes
Risk treatment decisions	No one fights for software security (RTD2)	-	Unclear responsibilities for sw security. Architects lack interest in security and do not take responsibility for security.	-
	Legal requirements dictate specific security measures - creates tension (RTD3)	Legal requirements and interaction with legal department	Balancing security with other needs.	-
Risk treatment follow up	Most trust the vendors and the developers (RTFU1)	-	Rely on contractors to deal with security – limited follow up. CISOs do not have resources to follow up on sw security.	Monitoring of changes in risk related to sw development is dependent on developer interest.
	Time pressure results in security requirements being postponed (or even dropped) (RTFU3)	-	Need for progress to fulfil contract Time pressure Agile development	-
Risk communication	Silo structure (RC2)	-	-	Legal expertise and security expertise in the organisations does not necessarily benefit development projects. Limited involvement by security people in the projects.

as a driver for security tool adoption, as well as a hindrance, depending on the circumstances. Only six of 17 security teams interacted often with developers. The other security teams focused more on operational security and was often found to be not helpful in development. When the security team interacted with developers, this caused more security activities to be performed by the developers due to social pressure, and in turn developers felt more responsible for the security of their code. The opposite effect was found in a company where only certain individuals were given training in use of security tools.

Involvement can be done and supported in many ways, e.g. through routines and by creating meeting places. In any case, it is important to ensure the voices of legal experts, security experts, and potentially other types of stakeholders related to security, is heard at key decision points in the projects to ensure that also their concerns are taken into account in the decisions on what risk should be accepted. Trade-offs will have to be made, but these trade-offs should be made based on awareness of the potential consequences of the choices available to the project.

## 5.2. Risk Perception and Competence

In the organisations studied, it seems clear that confidentiality is what is mainly considered when thinking about the needs for security in the projects. We found little awareness in the data that security is also about integrity and availability. As can be seen from Table 6, many of the obstacles when it comes to software security in the organisations is related to risk awareness; stakeholders have the opinion that security is not needed because the systems are only to be used internally or only contain open data. These are clearly factors that reduce the risk. However, it should not automatically lead to the conclusion that there is no need to consider security for such systems. Some security breaches are performed by insiders (Jang-Jaccard & Nepal, 2014). Some systems that start out as internal, are later wrapped to have a web interface etc. Open data may have requirements when it comes to integrity and availability. In addition, systems that only process open or otherwise insensitive data might still be





Table 6. Empirical basis for discussion and recommendations on risk perception and competence

Area	Main Finding	Relevant Triggers	Relevant Obstacles	Notes on Adoption
<i>Risk analysis</i>	-	Security considered part of what made the project technically interesting and challenging.	Systems not directly connected to the internet. Systems not used by externals. Open data. Lack of security knowledge among developers Procurers lack awareness about security. Security is mainly considered a technical issue. Lack of awareness of security implications.	-
<i>Risk treatment decisions</i>				
<i>Risk treatment follow up</i>				
<i>Risk communication</i>	Lack of training in software security risks (RC1)	-	No/little training in sw security	Training in sw security is almost non-existent in the organisations.

used as a step in an attack, which eventually gains access to other systems in the company. Thus, the organisations need to consider a broader set of security properties in their evaluations of security needs.

To improve security awareness, there is a need to do something about the current lack of training in software security among all stakeholders that are somehow involved in development, it being developers, software architects or procurers. This does not necessarily mean that all developers must attend a software security course (though this will surely have its benefits). Participation in risk analysis is an excellent way of increasing both security knowledge and awareness (Wheeler, 2011). Evaluations of Protection Poker (Williams et al., 2010) show that using a collaborative technique for discussing software security risks in the whole team raises awareness on security and spreads knowledge on security within the team. This corresponds to the experiences reported by Kongsli (Kongsli, 2006) on using misuse stories and automatic testing of security in the development of web applications. Effects experienced included increased security awareness in the team, raised collective ownership of security issues, and moving security activities such as system hardening and penetration testing to earlier iterations. However, to effectively raise awareness while also achieving a risk analysis with good quality, there is a need to include at least one person that is knowledgeable about software security. People with such competence thus need to be made available to the development projects to be used as a resource in security activities throughout the project. This may involve the need to train persons to fulfil this role, if such competence is not already available in the organisations. As also stated above, it is important to note that information security professionals do not automatically fit this role, as they may not have the necessary competence to understand the specifics of software development (van Wyk & McGraw, 2005).

### 5.3. Practical Ways of Doing Risk Analysis

In the organisations and projects studied, the work on software security does not seem to be risk centric, with a few exceptions (development project B (sub-study 2) and possibly a few of the organisations in the swsec maturity sub-study (sub-study 3)). Instead, organisations take a more compliance-based approach to security, with legal requirements as a main driver for security requirements and risk assessments. A compliance-based approach comes with its benefits, as was also discussed in the IS n&p sub-study (sub-study 1) in the context of organisational security work (not software security). In general, the participants in the focus groups did not agree on what would be most beneficial; a risk



based or a compliance-based approach to information security. It was pointed out that a risk-based approach requires more competence in order to be confident that major risks are taken care of. In cases where such competence is lacking, a checklist-based approach or an approach that mainly is based on legal requirements may result in better security. Legal requirements can then be considered a specification of a minimum security level that all organisations have to comply with.

Although a compliance-based approach is easier to apply, a large amount of literature and standards recommend the risk centric approach (Caralli et al., 2007; Chandra, 2008; Howard & Lipner, 2006; ISO/IEC, 2011; McGraw, 2006; McGraw et al., 2016; NIST, 2010; Wheeler, 2011). We would like to point out three main disadvantages with a compliance-based approach to security (based on Wheeler (2011)). First, there is no one-size-fits-all when it comes to security. Different organisations and software have different needs for security. If the security work is solely based on recommendations from checklists, one will likely improve security, but not in the most cost-effective way. In addition, one is not confident that the most important measures for this particular software is addressed sufficiently. Second, security threats are changing fast, thus checklists can be quickly outdated. In order to adequately react to changing threats, there is a need for competence and awareness beyond what you get from checklists. Third, with checklists, those responsible for addressing risks as well as other stakeholders are not trained in considering business needs and the role of security in fulfilling those. Participating in risk analysis is a great way to increase competence and awareness about security. Awareness of business needs is essential to make good decisions on what security measures to take and to ensure security is considered important by managers.

Based on literature and based on the findings from this study we thus recommend moving towards a more risk centric approach to software security. But in doing this the organisations and projects need improved ways to assess risk in the development projects.

Table 7 gives an overview of the main results from the study that we base these recommendations on. The studied organisations' current approaches to software security do not seem to give adequate confidence that software security is addressed sufficiently, as current software security efforts seem largely to be arbitrary, late and error driven (RTD1). Taking into account the limited testing and security review practiced (RTFU2), it is likely that many security issues go undetected with today's

**Table 7. Empirical basis for discussion and recommendations on practical ways of doing risk analysis**

Area	Main Finding	Relevant Triggers	Relevant Obstacles	Notes on Adoption
<i>Risk analysis</i>	Risk analysis practices vary greatly among organisations (RA1)	Legal requirements New products Project with obvious security needs	-	For risk analysis, see RA1-3. Very low adoption of threat modeling. -
	Legal requirements are a driver for performing risk analysis (RA2)			
	Risk analyses are often not centred on software security issues (RA3)			
<i>Risk treatment decisions</i>	Arbitrary, late and error driven (RTD1)	Detection of errors made	-	Some security requirements are made, but may come late in the process.
<i>Risk treatment follow up</i>	Limited security testing or review (RTFU2)	-	-	-
<i>Risk communication</i>	-	-	-	-

practice. Though errors were a trigger for security activities also in development project B where security was followed up in a more continuous and structured manner throughout the project, the effect of errors as a trigger for security activities seem to have been stronger in development project A where security was not given as much attention. Legal requirements were an important trigger in development project B, but in project B this led to a more risk centric approach to security with adoption of activities that could be considered part of risk management (see Table 3).

Current approaches to risk analysis in the studied organisations seem however to be inadequate for software security (RA1-3). First, it is not clear when risk analysis should be performed. Second, the analyses that are done seem not to be that useful for software security work. Based on this, it seems that the organisations could benefit from lightweight processes for determining the need for security in a project, to determine what level of risk analysis is needed as well as other security activities. Such a lightweight process need to take into account the full security properties, not only confidentiality and legal requirements. Additionally, there is a need for risk analysis methods that fit software projects and that can be done in a more continuous manner throughout the project and at a level useful for development. Protection Poker (Williams et al., 2010) is one example of a risk assessment technique tailored to agile development and that would potentially fit the needs of these organisations, and that has positive evaluation results in a real development setting. However, there are not many other such techniques to choose from. Instead companies are left with adapting more general risk assessment techniques to fit the needs of their agile development projects. More research is needed in how risk assessment can be done efficiently and effectively in agile projects.

#### 5.4. Threats to Validity

The discussion of threats to validity of this study is based on the recommendations of Cruzes and ben Othmane regarding threats to validity in empirical software security research (Cruzes & ben Othmane, 2017). It is important to highlight that qualitative studies such as the one that we performed rarely attempt to make universal generalisations. Instead, they are more concerned with characterising, explaining, and understanding the phenomena in the contexts under study. Cruzes and ben Othmane base their recommendations on Lincoln and Guba (1985), that substituted reliability and validity with the parallel concept of trustworthiness. Trustworthiness again consists of four aspects: credibility, transferability, dependability, and confirmability, with credibility as an analogy to internal validity, and transferability as an analogy to external validity.

Credibility refers to “the quality of being convincing or believable, worthy of trust” (Cruzes & ben Othmane, 2017), and dependability refers to “stability and reliability of data over time and conditions” (Cruzes & ben Othmane, 2017). The credibility and dependability of this study are closely related, and highly linked to study design decisions, in particular decisions regarding scope and depth of the study. In the following we discuss three main design decisions made and their impact on validity, namely the decision to study several organisations at a high level instead of one or a few organisations at a more detailed level, the decision to gather the perspectives of roles outside the development teams, and the decision to have the study organised as three sub-studies spanning two years.

In this study, we have studied 23 organisations, but these have not been studied in detail. We rely on self-reporting of practices, and thus on the people we talk with adequately reporting both practices and challenges. In the focus groups as well as in the interviews we got the impression that the people we talked to were honest about their practices, also telling about challenges they faced. However, we have not aimed to check that what they told us was in fact true using additional empirical sources. Usually, only one person from each organisation was interviewed, thus we only got one individual’s perspective on their software security work. We could have chosen to go more in depth in the organisations, including more peoples’ perspectives, but this would have come at the cost of the number of organisations we would have the capacity to study. Studying as many as 23 organisations from the same sector makes us able to understand the practices and challenges of this sector as a whole, not only that of individual organisations.

In the IS n&p sub-study (sub-study 1) and in the swsec maturity sub-study (sub-study 3), most of the people we talked to were not deeply involved in development, but rather had roles related to network security or information security in the organisation. One reason for this study design decision is our research question, where we take a more holistic approach to understanding risk centric practices to software security, including also the organisational aspect. This emphasis on the opinions of security people outside of the development projects is however a limitation, as we may risk to not adequately understand the actual practices of the development projects. By studying two development projects in more detail, we overcome some of this limitation. However, it is important to note that by collecting the viewpoints of security people, also outside development, is important because it corresponds to the way software security is currently handled in these organisations. When in the swsec maturity sub-study we asked to talk with those responsible for software security, we were directed to information security people in many cases. If we had decided to only study the projects, we would have missed the important perspective of the security people and their interaction with the projects when it comes to software security.

The study performed in order to do this mapping consists of three sub-studies performed over two years. The study does not aim to identify changes that may have happened during this time. However, we are not aware of any major external factors that would impact the software security work during these two years. The focus in the three sub-studies are not the same but differ in what is studied and who is used as informants. This is a strength in the way that the software security practices in the sector is studied from different angles. However, none of the studies study risk centric activities exclusively. Rather, the overall practices and challenges are aimed captured. As we did not bring with us a list of risk centric activities to look for in the companies, we may have missed some of their risk centric activities. However, studying the organisations strictly through the lens of such a list of risk centric activities could result in overlooking practices that we had not included in the list beforehand, thus obscuring the organisations' practices and approach to software security.

*Transferability* of study results refers to "the degree to which the results of the qualitative research can be generalized or transferred to other contexts or settings. It depends on the degree of similarity between sending and receiving contexts" (Cruzes & ben Othmane, 2017). In this case, it should not be done without taking into account the particular context of this study; public organisations in Norway. Public organisations may behave differently than private software companies in some respects. Although private companies act as contractors to the studied public organisations, the organisations themselves have different goals than what is common for private companies. The high emphasis on legal compliance found in this study is an example of a factor that can be stronger because of the sector and the type of systems developed. However, with the upcoming enforcement of the General Data Protection Regulation (GDPR) legal compliance will most likely become more important for software companies in general.

*Confirmability* refers to "neutrality; that is, findings must reflect the participants' voice and conditions of the inquiry, and NOT the researcher's bias, perspective, or motivations" (Cruzes & ben Othmane, 2017). In the three sub-studies, several researchers have been involved in data collection, and no one researcher has taken part in all data collection. This is both a strength and a weakness. By having several researchers involved, any preconceptions of one individual researcher have less impact on the data collection. However, with several researchers involved there is the challenge of coordinating these researchers to ensure the data collection is consistent among the different companies. This was especially important in the swsec maturity sub-study, where phone interviews were done by several researchers individually, and in this sub-study we took measures to ensure the involved researchers had a similar understanding of key concepts used in the interview guide and of the goal of the study. In the other two sub-studies, data collection was done by the same researchers throughout the whole sub-study. Richness of data has been ensured through recording of interviews in sub-studies 1 and 2. In sub-study 2, the data analysed was the transcribed interviews, while in sub-study 2 the recording was used to enrich notes taken during the focus group sessions. These notes, including initial conclusions

from the focus group, was sent to the focus group members for comments a short time after the focus group. In sub-study 3, richness of data was ensured by having questionnaires that was followed up by an interview. Besides, we have created mind-maps of the data for abstraction of the results, where all results can be traced back to the original source of information.

## 6. CONCLUSION

This study of software security practices in public organisations, with a mapping of their risk centric activities, has revealed that software security practices were not mainly based on an assessment of software security risks, but rather driven by compliance. Their practices could also in many cases be characterised as arbitrary, late and error driven, with limited follow up on any security issues throughout the development projects. We have identified a need for: more practical ways of doing risk analysis; improved risk perception and competence; and clearer responsibilities and improved stakeholder cooperation.

We recommend that organisations move towards a more risk centric approach to software security, as the current compliance-based approach does not give adequate confidence that important security issues are addressed sufficiently. In doing this, organisations would benefit from lightweight processes for determining the need for security in a project, making sure the full security properties are considered. Additionally, there is a need for risk analysis methods that fit agile software projects, and that can be done in a more continuous manner. Key stakeholders in the development project should be involved in risk analysis, to increase awareness of security. Responsibility for software security in a project should be clearly assigned, and should preferably be given to someone close to the development. However, in addition to assigning responsibility, there is a need to arrange for legal and security experts to have understanding of and interaction with development projects. When decisions are made that impact risk acceptance, there should be routines in place to make sure experts on legal and security issues have a chance to share their perspectives on the issue.

## ACKNOWLEDGMENT

This work has been supported in part by the SoS-Agile: Science of Security in Agile Software Development project, funded by the Research Council of Norway (grant number 247678). Two of the sub-studies (sub-study 1 and 3) have been funded by the Norwegian Agency for Public Management and eGovernment (Difi). We would like to thank all the company and project representatives that participated in the study. We would also like to thank Tor Erlend Fægri and Svein Hallsteinsen who performed the interviews in the second sub-study, and Karin Bernsmed who performed some of the interviews in the third sub-study. Thanks to Prof. Guttorm Sindre for input on the contents of the paper.

## REFERENCES

- Baca, D., Boldt, M., Carlsson, B., & Jacobsson, A. (2015, August 24-27). A Novel Security-Enhanced Agile Software Development Process Applied in an Industrial Setting. *Paper presented at the 2015 10th International Conference on Availability, Reliability and Security*.
- Bannerman, P. L. (2008). Risk and risk management in software projects: A reassessment. *Journal of Systems and Software, 81*(12), 2118–2133. doi:10.1016/j.jss.2008.03.059
- Beck, K. (2000). *Extreme programming explained: embrace change*. Addison-Wesley Longman Publishing Co., Inc.
- ben Othmane, L., Angin, P., Weffers, H., & Bhargava, B. (2014). Extending the Agile Development Process to Develop Acceptably Secure Software. *IEEE Transactions on Dependable and Secure Computing, 11*(6), 497-509. doi:10.1109/tdsc.2014.2298011
- Caralli, R. A., Stevens, J. F., Young, L. R., & Wilson, W. R. (2007). *OCTAVE Allegro: Improving the Information Security Risk Assessment Process (CMU/SEI-2007-TR-012 ESC-TR-2007-012)*. Software Engineering Institute at Carnegie Mellon University.
- Chandra, P. (2008). Software assurance maturity model. Retrieved from Cruzes, D. S., & ben Othmane, L. (2017). Threats to Validity in Empirical Software Security Research. In L. ben Othmane, M. G. Jaatun, & E. Weippl (Eds.), *Empirical Research for Software Security: Foundations and Experience* (pp. 277-302). CRC Press.
- Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. *Computer, 34*(11), 131–133.
- Cybenko, G. (2006). Why Johnny Can't Evaluate Security Risk. *IEEE Security and Privacy, 4*(1), 5. doi:10.1109/MSP.2006.30
- Deleersnyder, S., Win, B. D., & Glas, B. (2017). *Software Assurance Maturity Model - How To Guide - A Guide to Building Security Into Software Development*. Retrieved from [https://github.com/OWASP/samm/blob/master/v1.5/Final/SAMM\\_How\\_To\\_V1-5\\_FINAL.pdf](https://github.com/OWASP/samm/blob/master/v1.5/Final/SAMM_How_To_V1-5_FINAL.pdf)
- Dingsøy, T., Moe, N. B., Fægri, T. E., & Seim, E. A. (2017). Exploring software development at the very large-scale: A revelatory case study and research agenda for agile method adaptation. *Empirical Software Engineering*. doi:10.1007/s10664-017-9524-2
- Eclipse. (2016). Eclipse Process Framework (EPF). Retrieved from <http://www.eclipse.org/epf/>
- Fenz, S., & Ekelhart, A. (2010). Verification, validation, and evaluation in information security risk management. *IEEE Security and Privacy, 2*(2): 58–65.
- Fitzgerald, B., & Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software, 123*(Suppl. C), 176–189. doi:10.1016/j.jss.2015.06.063
- Gerber, M., & Von Solms, R. (2005). Management of risk in the information age. *Computers & Security, 24*(1), 16–30. doi:10.1016/j.cose.2004.11.002
- Hijazi, H., Khdour, T., & Alarabeyyat, A. (2012). A Review of Risk Management in Different Software Development Methodologies. *International Journal of Computers and Applications, 45*(7), 8–12.
- Howard, M., & Lipner, S. (2006). *The Security Development Lifecycle: A Process for Developing Demonstrably More Secure Software* (Vol. 2016). Microsoft Press.
- ibbs, C. W., & Kwak, Y.-H. (2000). Assessing project management maturity. *Project Management Journal, 31*(1), 32–43.
- Islam, S., Mouratidis, H., & Weippl, E. R. (2014). An empirical study on the implementation and evaluation of a goal-driven software development risk management model. *Information and Software Technology, 56*(2), 117–133. doi:10.1016/j.infsof.2013.06.003
- ISO/IEC. (2011). ISO/IEC 27005: 2011 Information technology—Security techniques—Information security risk management.

- Jaatun, M. G., Cruzes, D. S., Bernsmed, K., Tøndel, I. A., & Røstad, L. (2015). *Software Security Maturity in Public Organisations*. In *Information Security* (pp. 120–138). Springer.
- Jang-Jaccard, J., & Nepal, S. (2014). A survey of emerging threats in cybersecurity. *Journal of Computer and System Sciences*, 80(5), 973–993. doi:10.1016/j.jcss.2014.02.005
- Jourdan, Z., Rainer, R. K., Marshall, T. E., & Ford, F. N. (2010). An Investigation Of Organizational Information Security Risk Analysis. *Journal of Service Science*, 3(2), 10.
- Junior, I. H. F., Azevedo, R. R. d., Moura, H. P. d., & Silva, D. S. M. d. (2012, August 27-30). Elicitation of Communication Inherent Risks in Distributed Software Development. *Paper presented at the 2012 IEEE Seventh International Conference on Global Software Engineering Workshops*. doi:10.1109/ICGSEW.2012.18
- Kongsli, V. (2006). Towards agile security in web applications. *Paper presented at the Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*.
- Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic inquiry* (Vol. 75). Sage.
- McGraw, G. (2004). Software security. *IEEE Security and Privacy*, 2(2), 80–83. doi:10.1109/MSECP.2004.1281254
- McGraw, G. (2006). *Software security: building security in* (Vol. 1). Addison-Wesley Professional.
- McGraw, G., Miguez, S., & West, J. (2013). *Building Security In Maturity Model (BSIMM-V)*. Retrieved from <http://bsimm.com>
- McGraw, G., Miguez, S., & West, J. (2016). *Building Security In Maturity Model (BSIMM7)*. Retrieved from <http://bsimm.com>
- Microsoft. (2009). *Security Development Lifecycle for Agile Development*. Retrieved from <http://www.microsoft.com/en-us/SDL/Discover/sdlagile.aspx>
- Nelson, C. R., Taran, G., & de Lascrain Hinojosa, L. (2008). Explicit Risk Management in Agile Processes. In P. Abrahamsson, R. Baskerville, K. Conboy, B. Fitzgerald, L. Morgan, & X. Wang (Eds.), *Agile Processes in Software Engineering and Extreme Programming: 9th International Conference, XP 2008, Limerick, Ireland, June 10-14* (pp. 190-201). Berlin: Springer.
- NIST. (2010). *Guide for Applying the Risk Management Framework to Federal Information Systems - A Security Life Cycle Approach* (Special Publication 800-37). Retrieved from <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r1.pdf>
- Nyffjord, J., & Kajko-Mattsson, M. (2008). Integrating Risk Management with Software Development: State of Practice. *Paper presented at the International MultiConference of Engineers and Computer Scientists*, Hong Kong. Retrieved from [http://www.iaeng.org/publication/IMECS2008/IMECS2008\\_pp878-884.pdf](http://www.iaeng.org/publication/IMECS2008/IMECS2008_pp878-884.pdf)
- Odzaly, E. E., Greer, D., & Stewart, D. (2017). Agile risk management using software agents. *Journal of Ambient Intelligence and Humanized Computing*. doi:10.1007/s12652-017-0488-2
- Oppliger, R. (2015). Quantitative Risk Analysis in Information Security Management: A Modern Fairy Tale. *IEEE Security and Privacy*, 13(6), 18–21. doi:10.1109/MSP.2015.118
- Oyetoyan, T. D., Jaatun, M. G., & Cruzes, D. S. (2017). A Lightweight Measurement of Software Security Skills, Usage and Training Needs in Agile Teams. *International Journal of Secure Software Engineering*, 8(1), 27. doi:10.4018/IJSSE.2017010101
- Poller, A., Kocksch, L., Türpe, S., Epp, F. A., & Kinder-Kurlanda, K. (2017). Can Security Become a Routine?: A Study of Organizational Change in an Agile Software Development Group. *Paper presented at the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, Portland, OR. doi:10.1145/2998181.2998191
- Raz, T., Shenhar, A. J., & Dvir, D. (2002). Risk management, project success, and technological uncertainty. *R & D Management*, 32(2), 101–109. doi:10.1111/1467-9310.00243
- Rhee, H.-S., Ryu, Y. U., & Kim, C.-T. (2012). Unrealistic optimism on information security management. *Computers & Security*, 31(2), 221–232. doi:10.1016/j.cose.2011.12.001
- Stewart, D. W., & Shamdasani, P. N. (2014). Focus groups []: Sage Publications.]. *Theory into Practice*, 20.

Sulaman, S. M., Weyns, K., & Höst, M. (2013). A review of research on risk analysis methods for IT systems. *Paper presented at the 17th International Conference on Evaluation and Assessment in Software Engineering*. doi:10.1145/2460999.2461013

Tavares, B. G., da Silva, C. E. S., & de Souza, A. D. (2017). Risk management analysis in Scrum software projects. *International Transactions in Operational Research*. doi:10.1111/itor.12401

Tøndel, I. A., Line, M. B., & Johansen, G. (2015). Assessing information security risks of AMI: What makes it so difficult? *Paper presented at the 1st International Conference on Information Systems Security and Privacy 2015*, Angers, France.

van Wyk, K. R., & McGraw, G. (2005). Bridging the gap between software development and information security. *Security & Privacy, IEEE*, 3(5), 75–79. doi:10.1109/MSP.2005.118

Wheeler, E. (2011). *Security Risk Management* (1st ed.). Boston: Syngress.

Williams, L., Meneely, A., & Shipley, G. (2010). Protection Poker: The New Software Security “Game”. *IEEE Security and Privacy*, 8(3), 14–20. doi:10.1109/MSP.2010.58

Xiao, S., Witschey, J., & Murphy-Hill, E. (2014). Social influences on secure development tool adoption: why security tools spread. *Paper presented at the Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, Baltimore, MD. doi:10.1145/2531602.2531722

APPENDIX

Mind Maps of Findings

Figures 4-7 provide an overview of the main findings from the study, as well as which sub-studies the findings come from.

Figure 4. Overview of key findings on risk analysis

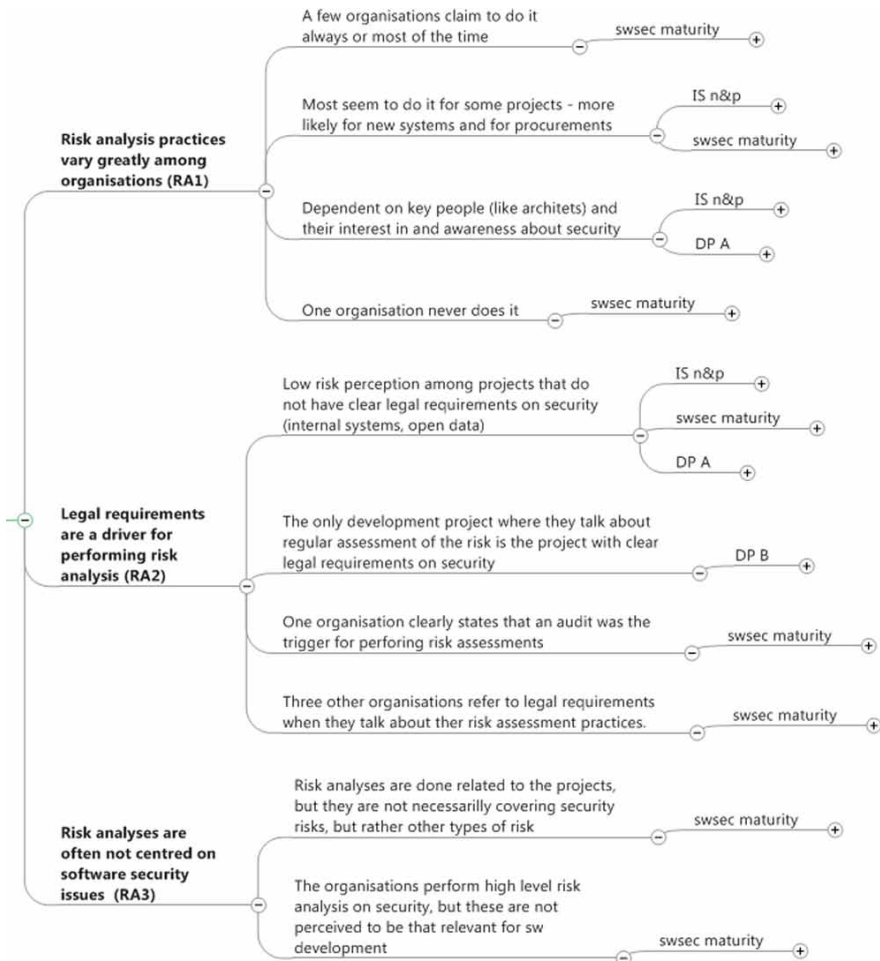




Figure 5. Overview of key results on risk treatment decisions

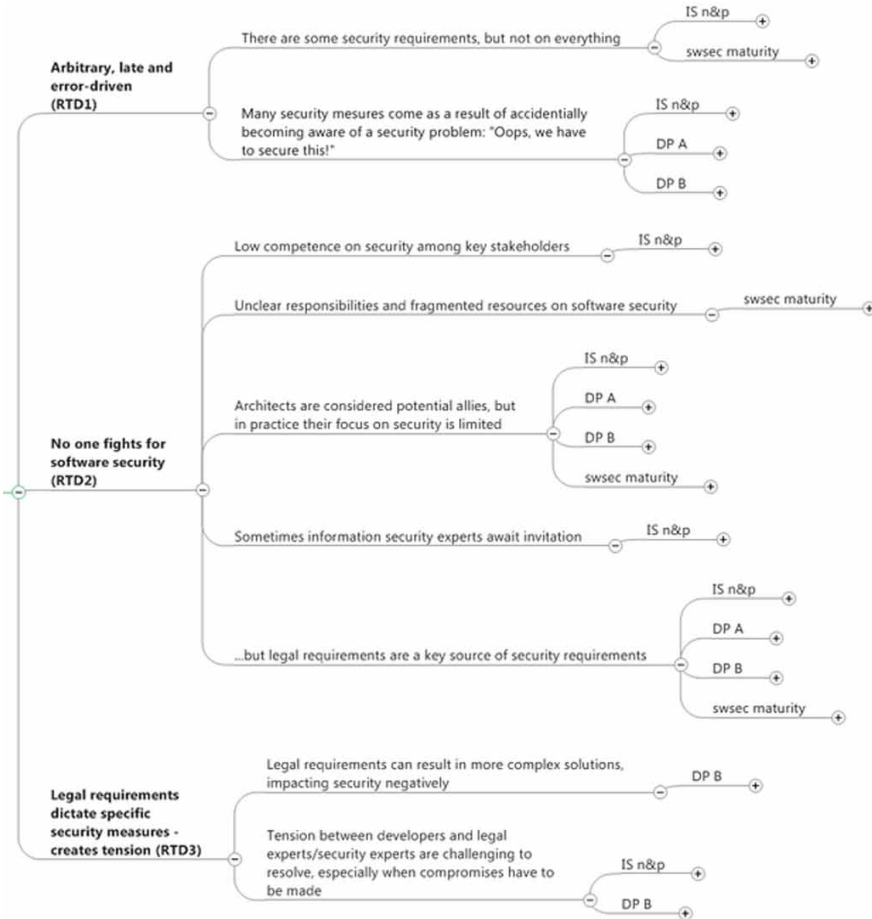


Figure 6. Overview of key results on risk treatment follow up

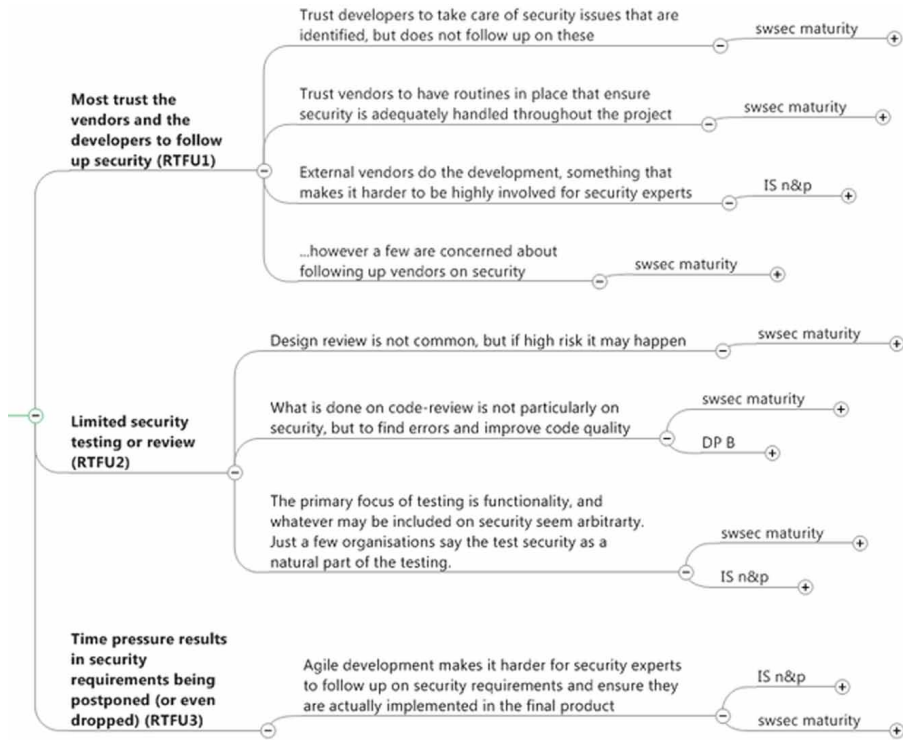
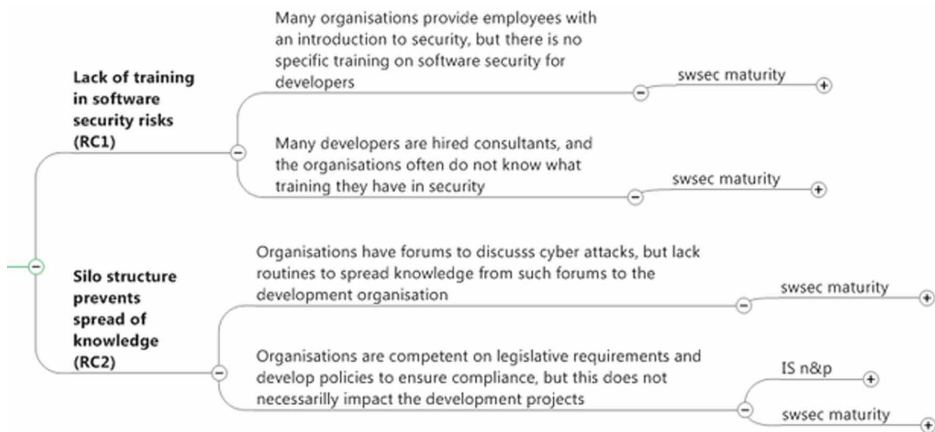


Figure 7. Overview of key results on risk communication



*Inger Anne Tøndel is a PhD candidate at the Department of Computer Science at the Norwegian University of Science and Technology (NTNU) and a Research Scientist at SINTEF Digital. She got her master's degree in Telematics from NTNU in 2004. Her research interests include software security, information security risk management, cyber insurance and cyber security in smart grids.*

*Martin Gilje Jaatun is a Senior Scientist at SINTEF Digital and an Adjunct Professor at the University of Stavanger. He graduated from the Norwegian Institute of Technology (NTH) in 1992 and received the Dr.Philos. degree from the University of Stavanger in 2015. Previous positions include scientist at the Norwegian Defence Research Establishment (FFI), and Senior Lecturer in information security at the Bodø Graduate School of Business. His research interests include software security, security in cloud computing, and security of critical information infrastructures. He is vice chairman of the Cloud Computing Association (cloudcom.org), vice chair of IEEE TCCLD, and a Senior Member of the IEEE. He is also an IEEE Cybersecurity ambassador, and Editor-in-Chief of the International Journal of Secure Software Engineering.*

*Daniela Cruzes is a researcher scientist at SINTEF. Previously, she was adjunct associate professor at the Norwegian University of Science and Technology (NTNU). She worked as a researcher fellow at the University of Maryland and Fraunhofer Center for Experimental Software Engineering-Maryland. Dr. Daniela Cruzes received her PhD in experimental software engineering from the University of Campinas - UNICAMP in Brazil in 2007. Her research interests are empirical software engineering, research methods and theory development, synthesis of SE studies, software security, software testing and agile and DevOps.*

*Nils Brede Moe works with software process improvement, intellectual capital, and agile and global software development as a senior scientist at SINTEF. His research interests are related to organizational, socio-technical, and global/distributed aspects. His publications include several longitudinal studies on self-management, decision making, innovation, and teamwork. He has co-edited the books Agile Software Development: Current Research and Future Directions and Agility Across Time and Space: Implementing Agile Methods in Global Software Projects. His thesis was, From Improving Processes to Improving Practice - Software Process Improvement in Transition from Plan-driven to Change-driven Development. He holds an adjunct position at the Blekinge Institute of Technology in Sweden.*

## **Paper B: ‘IT Security Is From Mars, Software Security Is From Venus’**

©2020 IEEE. Reprinted, with permission, from Inger Anne Tøndel, Martin Gilje Jaatun, and Daniela Soares Cruzes, IT Security Is From Mars, Software Security Is From Venus, IEEE Security & Privacy, July/August 2020.

Included here is the version of the article that was accepted by IEEE for publication. The final and published version is the reviewed and accepted article, with copy-editing, proofreading and formatting added by IEEE. That is the version that appears in IEEE Xplore and can be found following the reference [28].

**B**

# IT security is from Mars, software security is from Venus

Inger Anne Tøndel, Martin Gilje Jaatun, Daniela Soares Cruzes

**Abstract: In smaller software companies, the divide between IT security and software security can result in software security not being prioritized. A formal security champion role in the development team and collaborative risk-based security activities are potential ways to reduce this divide and bring a more proactive approach to software security.**

More than ten years ago, van Wyk and McGraw [1] called out for aligning information security and software development. At that time, there was a disconnect between security and development that led to software being developed without any understanding of technical security risk, and thus software with security weaknesses that should have been avoided. Even though the software security landscape has changed a lot in the past ten years, with increasing exposure of software and growing attention to security issues, this disconnect is still present in software companies (Table 1).

We have studied software security practices and challenges in 23 public organizations in Norway [2] through interviews with employees having various roles related to security in these organizations. This includes CISOs or other personnel with IT security or network security roles in the organizations, as well as software architects. The organizations we studied vary in size, but most can be considered small or medium sized companies (SMEs). The public organizations additionally varied in how much software was internally developed and what role the organization itself had in the development; whether they did development in-house, hired external developers, acquired bespoke software from vendors, or a combination of the above. The majority had their own software developers, who often worked together with hired external developers.

In the study we identified various ways in which the disconnect between security and development is still prominent in these smaller organizations, as well as reasons why this is so. In the following we explain these reasons further, and provide some suggestions for how to move forward from here. We found that to the extent that IT security professionals are involved in the development, this involvement is not strategic, and the man-hours put into this interaction is very limited. Additionally, a lot of the good work that is done on IT security and network security in the organization, does not seem to influence software development – it rather is seen as irrelevant for the development. This goes for business-level information security risk analysis as well as for penetration testing.

Table 1. To what extent do the organizations and projects follow the recommendations of van Wyk and McGraw on information security professionals' involvement in software development projects?

Recommendation van Wyk and McGraw	IT security professionals' practice	Software development professionals' practice
<p><b>Abuse cases:</b> Security professionals have knowledge of attacks, and should participate together with developers in creating abuse cases</p>	In general, IT security practitioners are <i>not</i> involved in creating abuse cases, but they may have conversations with developers about threats and security requirements.	Only very few create abuse cases.
<p><b>Business risk analysis:</b> Information security professionals know security impacts first hand for similar business applications, and can thus provide answers to questions on incident costs.</p>	Perform overall risk analysis for the whole organization, but these are often considered by developers to not be relevant for the development projects.	Several organizations do risk analysis related to development projects, but these do not necessarily cover security risks. Only a few have clear routines to do software security risk analysis related to the development projects.
<p><b>Architectural risk analysis:</b> A security analyst that is also a technology expert (covering application, underlying platform, frameworks, languages, etc.) can provide important perspectives on risks, weaknesses and mitigation strategies.</p>	-	When security architects are involved in the projects, these may evaluate risk related to architectural decisions and follow up on security principles. This is however seldom done unless security is a clear priority.
<p><b>Test planning:</b> Risk based testing scenarios would benefit from experiences of incident handlers. Security professionals are good at "thinking like an attacker".</p>	-	Testing mainly covers functionality.
<p><b>Code review:</b> This step is best left in the hands of the development organization.</p>	-	Code review is commonly performed, but related to code quality in general (no specific focus on security).
<p><b>Penetration testing:</b> This is usually the domain of information security and incident handling organizations, but for software development a more inside -&gt; out approach should be taken</p>	Several organizations do penetration testing, but not necessarily directed at the software they develop. Initiatives for penetration testing often come from outside the development organization.	A few do penetration testing at main release, or if they suspect major security issues.

Recommendation van Wyk and McGraw	IT security professionals' practice	Software development professionals' practice
<p><b>Deployment and operations:</b> Information security expertise can help safely setting up the application in a secure operational environment; access controls, event logging and monitoring, etc.</p>	<p>Security is in general a much higher priority and concern in operation (network security). This part of the organization usually has routines to stay updated on attacks and security risk.</p>	<p>Different culture among developers and operations when it comes to security. This may lead to frictions; e.g. developers believe operations put up too many hindrances.</p>

So, where does it go wrong? The evidence we have collected points to three main reasons why software security is not given priority, as summarized in Table 2. These reasons concern both the IT security and development tribes. In the following we go into each of these reasons, explain the challenges we identified and provide suggestions for moving forward.

Table 2: Key reasons why software security is not given priority, both among IT security professionals and in the development organization

Unclear responsibilities and expectations on software security	Risk perception	Lack of approaches that fit the software development daily activities
<ul style="list-style-type: none"> <li>No one is given explicit responsibility for software security</li> <li>Optimistic assumptions on competence and interest of developers and contractors on security</li> </ul>	<ul style="list-style-type: none"> <li>Software security not important for internal systems</li> <li>Security is about confidentiality</li> <li>Contractors and developers can be trusted</li> </ul>	<ul style="list-style-type: none"> <li>Approaches to security that worked for waterfall-based development do not work as well with agile</li> <li>IT security people not involved in sprint meetings or other key decision making points</li> </ul>

### Unclear responsibilities: Where does IT security stop and software security begin?

Commonly, information security is defined as safeguarding the confidentiality, integrity and availability of information, and IT security is broadly defined as information security in IT systems. In today's businesses, this information is in large part processed by software systems, thus software security is essential for information security. Information security management standards, such as ISO/IEC 27001, include controls on system acquisition, development and maintenance. It is therefore not a surprise that in the organizations, IT security personnel are often given some responsibility for software security.

McGraw defines software security as "the idea of engineering software so that it continues to function correctly under malicious attack" [3]. Table 3 provides an overall comparison between the information security and software security fields. The fields are clearly related. Still, there are major differences in the formal requirements to, and the organization of, the work. van Wyk and McGraw recommended that a fruitful cooperation between information security people and developers could help developers understand what they're up against and potential impacts on the business. To achieve such a fruitful cooperation, they recommended that information security professionals



should be involved in some of the software security touchpoints (Table 1). But van Wyk and McGraw were very clear that this required skills and initiative from the IT security side. Gaining the necessary understanding of software development in order to contribute with security in a meaningful way, and in a way that is respected by the developers, is non-trivial. This included understanding the craft of developing software, as well as what are the goals driving the development and its race towards faster time to market.

Table 3. IT security and software security compared

	ITSEC	SWSEC
<b>FORMAL RESPONSIBILITIES</b>	Place in the hierarchy; a formal role with responsibilities.	No formal position. Autonomous teams
<b>MATURITY</b>	Standards are adopted. Mature tools.	Standards not often used. Less mature tools.
<b>DRIVERS</b>	Risks; incidents; standards and legislation.	Requirements (legislative and customer demands); software vulnerabilities.
<b>RESTRICTIONS</b>	Costs-effectiveness; management buy-in.	Time to market.
<b>GENERAL MINDSET</b>	Sceptic and risk averse.	Optimistic – build things.
<b>SPEED</b>	Two paces: i) race against attackers (patch, update signatures, etc.) ii) ISMS – plan, do, check, act – longer cycles (e.g. risk analysis once a year)	Agile, DevOps, continuous delivery -> need to keep up with this pace.

### Software security rely on individual initiative

Most organizations in our study point to IT security people as the ones having responsibility also for software security. However, IT security people seem not to be highly involved in software development, and for them, their responsibility for software security is unclear. As a result, the responsibility is fragmented, and it is not possible to clearly hold anyone accountable for software security. Some IT security professionals stated that, in the end, the developers are responsible for their own part of the system, and that in their organization, other security activities and goals had been given priority. Consequently, software security had not been given attention. Many organizations seem to rely quite heavily on their contractors to take care of software security, and do not really follow up on them regarding security issues. They rely on contractors to identify security requirements, and assume that they have an overview of security risk and perform the right activities to address this risk.

IT security professionals may occasionally discuss security issues with developers, but they do not follow up on how this is dealt with in the development. IT security professionals view architects as important and potential allies in the software security work. However, in practice the architects seldom take on this role as a security ally. The architects often come from external contractors, and are thus mainly concerned with getting the job done, and following the product owner's orders. Interviewees talk about situations where the architect does not take responsibility for security, and instead points to the CISO or similar role. Since architects are considered to be a primary influencer on whether there is, e.g., performed design or architecture review related to security, software security currently relies on individual initiative and interest of security among the architects. On the plus side, since architects typically are seasoned developers with significant experience, it is likely

that they will have more software security knowledge than the average developer, as our studies indicate that software security knowledge is correlated with years of experience [4]. This means that architects should be well placed to fill a role in software security, but this responsibility needs to be assigned explicitly by management.

### The CISO as a change agent for software security?

For software security to gain momentum, someone needs to ask for more software security to drive change in practices. If assuming that the disconnect between IT security personnel and developers is the main reason why software security is not happening, one could say that the initiative for software security should come from the IT security side. This is also very much what van Wyk and McGraw build on in their article, providing recommendations for information security experts on how to become more involved in development. However, we don't see evidence that this is happening.

So, if neither the security nor the developer side is pushing for more software security, who should? As a general rule, management is key in driving change in organizations [5]. They are in a position to push security in the organization, either information security or software security. In the organizations studied there is a big difference in the awareness and push from managers on information security compared to software security. At the time of the first part of the study (2013), all the public organizations were required to implement an information security management system (ISMS). At the same time, large public development projects could run without software security being a main consideration, and without anyone being given clear responsibility for software security.

Though information security practitioners and developers often have a common technical background, the former rarely have strong development expertise. Thus, it is generally recommended that responsibility for software security should be assigned to someone from the development side. Based on data from the BSIMM study [6], the first step in a software security initiative should be the formation of a software security group (SSG), responsible for carrying out and facilitating software security in the organization. This group should ideally consist of software security people, alternatively developers that can be taught about security. The SSG should have people with deep coding skills as well as architects, and people with good communication skills. It has been stated that network security people usually fail in this type of role [6].

If looking at organization charts, CISOs (and other information security or network security people) are usually not located anywhere near the development organization. It has previously been shown how the silo structure of organizations can limit communication about and learning from cyber incidents [7]. Similarly, organizations may find that the brilliant information security competence they have in-house does not benefit the development at all. In the organizations studied, developers are not included in security forums that, e.g., discuss attacker trends and risks, and IT security people are only occasionally in interaction with the development organizations. In the organizations that rely quite heavily on external contractors for development, the linkage and proximity of the IT security and development people are an even bigger challenge.

### Security Champions can be the bridge between IT security and Software Security

Just telling software developers and IT security people that they need to play together has not been working – a decade's worth of empirical evidence tells us this. Another change agent is necessary, and we propose that this role can be filled by *software security champions* [8, 9]. As mentioned before, establishing a security champion program also needs to be management-driven. Developers with an above-average interest in software security need to be identified or hired, and care must be

taken that every team has at least one security champion. This will require management support and funding. However, this person must NOT be an external IT security expert – it is instrumental that the security champion is a developer, contributing to the development process and the quest toward "done". The BSIMM [6] also highlights the important role of the security champions, but uses the term "satellite" instead, suggesting that they are somewhat secondary to the SSG. We believe that for SMEs, it will be more beneficial to start with the software security champions, and possibly migrate to creating an SSG if and when the organization reaches the requisite size.

Once the security champions are in place, they can serve as the bridge between the CISO and the developers – their security knowledge should let them understand the "security-speak" of the CISO, and their developer chops and positive contributions to the fight against the windmills represented by the backlog should ensure a sympathetic ear among their fellow developers. For organizations that have an SSG, this just adds another layer to the organization chart, as shown in *Figure 1*. It is important that the security champions organizationally are placed in the same hierarchy as other developers, ultimately reporting to the Vice President of Development or similar role. We believe the same holds true for the SSG – even though it should have clear lines to the CISO, it is still part of the development organization

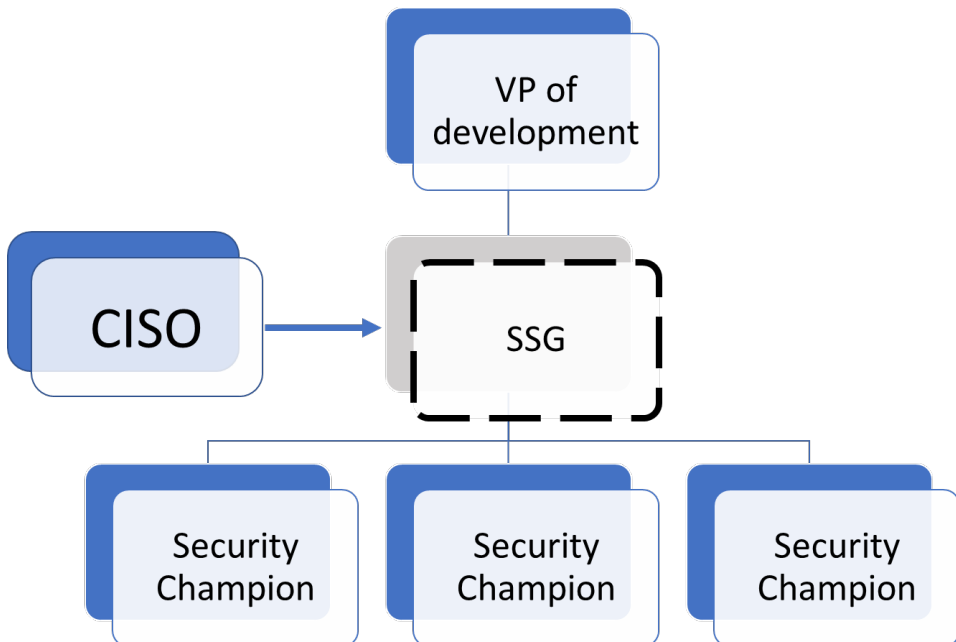


Figure 1: Shoehorning security into the organization chart

### Risk perception

Despite the common mantra in literature and in security circles that all security work should be risk based, we did not find much evidence that software security follows a risk-based approach in the organizations we studied. Instead the approach can be characterized as compliance-based or accidental.

The organizations in our study were up to date on legal requirements, network security and new risks in IT. This however does not mean that the software development initiated by these organizations benefited from that competence. Though the studied organizations often have forums and similar to follow up on the latest cyber security threats, none seem to have clear routines to inform developers about new or evolving threats. Only a few organizations have identified what their most important product is, or which kinds of attacks they are most afraid of related to their software.

When the systems under development are not to be open for external users, and thus not directly available on the Internet, security is not considered to be particularly important. This is the case for many of the systems developed by or for these organizations, and in general, security does not seem to be prioritised. Instead, efficiency (i.e., getting the job done) is considered the main priority both from those procuring and those developing the software. Additionally, the organizations seem to have significant trust in the software developers, including contractors, that they all have good intentions. Thus, they do not feel the need to have mechanisms in place to check for rogue code, and do external security tests, etc.

IT security personnel are not necessarily involved in making decisions on what level of security is needed for the project. Software security is more likely to be considered for development of new products, than for improvements of existing products, and if the need for security is rather obvious (e.g. health information), as this can spur the involvement of security experts in the project. If the need is not obvious, security may not be considered – unless something triggers a sudden jump in security attention.

The main trigger for security activities in the development projects are accidentally detected security vulnerabilities and legal requirements. In the one project we studied that had security as a high priority, this was solely because of strict legal requirements to the type of data the software should handle. Because of these legal requirements, security was given priority in the budget and security architects were included in the development teams. Legal requirements are additionally stated as a reason for doing risk analysis. In some cases, risk analysis is first performed after receiving audit remarks that this is lacking. For projects without a clear approach to security, security activities come a bit incidental and late (if at all).

### Risk centred activities as an antidote

As development organizations increasingly become aware of the need to address security during development efficiently and effectively, without hampering their agile approach to development, they need to make assessments on what type of security activities to include, i.e., what security activities pay off. We advocate that companies would benefit from taking a risk-based approach in their selection of software security activities, to ensure they are conscious about how much security and what type of security is most needed in their specific project. Making such decisions is challenging and requires security competence. However, there are many activities that development projects can adopt that make such decisions more available to the team, also in cases where they lack deep security knowledge.

The most often mentioned activity is Threat analysis/Threat modeling. In such activities the development teams can be supported by checklists or mnemonics such as STRIDE, or they can even utilize game-based approaches such as OWASP Cornucopia [10] and Microsoft EoP [11]. Alternatively, including developers in risk analysis activities can make these analyses more relevant to the team and increase the security awareness of those that participate. Protection Poker [12], a game-based approach to risk estimation, offers a way to easily integrate security risk analysis into agile development practices. These activities require little preparation, require relatively low effort to

perform, and contribute to building security awareness in the whole team. Essentially, they are structured ways to talk about security risk, and may be the little push that is required to spend more time on software security activities. Including IT security personnel in the activities can further improve the quality of the security discussions they foster, and increase awareness and understanding on both sides; making the development teams more knowledgeable and aware of security issues, and increase IT security personnel's understanding of the challenges in development.

## Security in Software Development Practices: Speed, Data, Ecosystems

It is clear that many IT security professionals and developers have conflicting goals. While the IT security people in general express a concern that software security may not be adequately handled in their organization, both groups explain that the main priority during development is functionality. Additionally, in cases where security requirements are identified early on, time pressure in the projects can cause security requirements to be postponed, even past deployment. With traditional waterfall development methods, all requirements were in the contract and had to be fulfilled. Now, important security decisions are made in short sprint meetings, localized in the scope of the sprint, and the impression is that whenever there is a conflict over time and budget in a sprint, security is sacrificed in order to realize functionalities. Most organizations we have studied are struggling to integrate security into the agile way of working. One participant explained that information security is not included until the end of the project, as information security personnel is not invited to participate before then, but at that point developers do not like it when she introduces new security requirements.

According to an article by Bosch [13], there are three key factors driving the future of software engineering:

- *speed*: continued success depends on the organization's ability to respond quickly to customer requests, changing market priorities, new competitors, etc.
- *data*: data collection is now cheap and easy, and organizations that are able to make smart and timely decisions based on collected data will benefit
- *ecosystems*: as a result of increased speed and data, companies will more frequently change their role and position in their ecosystem, thus organizations need to intentionally, proactively and effectively manage changing relationships

These drivers impact the whole organization, not only the development teams. For software development, there has long been a drive towards less documentation and faster delivery of working code, through agile development, DevOps, and continuous delivery. Where traditional organizations relied on functional organizational hierarchies, businesses now move towards cross-functional teams and self-management. CISOs and similar organizational roles need to find their way in an organization structure more centered on autonomous teams, where speed of decisions is key to maintain competitiveness. We find that this aligns well with the suggestion that the CISO exerts influence on a distributed band of security champions, possibly via the SSG, without removing their organizational ties to the development organization.

Agile is all about value and functionality; what you can show to the customer in the next sprint. Security tends to be considered a non-functional requirement, and with the main emphasis on functionality, such non-functional requirements are easily sacrificed on the altar of progress when backlog priorities are set. In case of security incidents, there is a need for rapid response, but the overall challenges are more towards getting management approval of needed, longer-term security investments. However, we expect that as organizations move toward DevOps and continuous

deployment, IT security will automatically need to be tighter integrated with development, and their horizons will shift accordingly.

It can be argued that Ops are in general more security aware, since they have to deal with daily intrusion attempts and manage firewalls, antivirus tools and intrusion detection systems. The DevOps mantra is "you build it – you run it", which implies that the developers will be much closer to the sharp end, also when an incident should happen. This also means that developers need to be involved in incident response drills, ultimately resulting in better response and quicker fixing of security bugs and flaws.

Along with the drive towards value and functionality fast, there is a growing attention to the fact that software needs to be developed with security in mind. Cyber incidents are visible in media, and this increases awareness among managers as well as customers. It is thus likely that in the future also non-security-critical software will need to consider security as an important quality attribute, and that customers will pose security requirements to the software they acquire. This in itself can increase the drive for security in the development organization without IT security people needing to take the role of a change agent for software security.

## Acknowledgements

This work was supported by the *Science of Security in Agile Software Development* project (SoS-Agile), funded by the Research Council of Norway (grant number 247678).

## References

- [1] K. R. van Wyk and G. McGraw, "Bridging the gap between software development and information security," *Security & Privacy, IEEE*, vol. 3, pp. 75-79, 2005.
- [2] I. A. Tøndel, M. G. Jaatun, D. S. Cruzes, and N. B. Moe, "Risk Centric Activities in Secure Software Development in Public Organisations," *International Journal of Secure Software Engineering (IJ SSE)*, vol. 8, pp. 1-30, 2017.
- [3] G. McGraw, "Software security," *IEEE Security & Privacy*, vol. 2, pp. 80-83, 2004.
- [4] T. D. Oyetoyan, M. G. Jaatun, and D. S. Cruzes, "A Lightweight Measurement of Software Security Skills, Usage and Training Needs in Agile Teams," *International Journal of Secure Software Engineering*, vol. 8, p. 27, January 2017.
- [5] H. Assal and S. Chiasson, "Security in the software development lifecycle," presented at the Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018), 2018.
- [6] G. McGraw, S. Miguez, and J. West, "Building Security In Maturity Model (BSIMM9)," Synopsys October 2018.
- [7] A. Ahmad, J. Hadgkiss, and A. B. Ruighaver, "Incident response teams—Challenges in supporting the organisational security function," *Computers & Security*, vol. 31, pp. 643-652, 2012.
- [8] V. Asthana, K. Beckers, M. Ifland, J. Martin, N. Ozmore, I. Tarandach, *et al.*, "Software: Security Takes a Champion," <http://safecode.org/wp-content/uploads/2019/02/Security-Champions-2019-.pdf> 2019.
- [9] A. Antukh. (2017). *Security Champions Playbook*. Available: [https://www.owasp.org/index.php/Security\\_Champions\\_Playbook](https://www.owasp.org/index.php/Security_Champions_Playbook)

- [10] M. Thompson and H. Takabi, "EFFECTIVENESS OF USING CARD GAMES TO TEACH THREAT MODELING FOR SECURE WEB APPLICATION DEVELOPMENTS," *Issues in Information Systems*, vol. 17, 2016.
- [11] A. Shostack, "Elevation of privilege: Drawing developers into threat modeling," in *2014 {USENIX} Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, 2014.
- [12] L. Williams, M. Gegick, and A. Meneely, "Protection poker: Structuring software security risk assessment and knowledge transfer," in *International Symposium on Engineering Secure Software and Systems*, 2009, pp. 122-134.
- [13] J. Bosch, "Speed, Data, and Ecosystems: The Future of Software Engineering," *Software, IEEE*, vol. 33, pp. 82-88, 2016.

## **Paper C: ‘Collaborative security risk estimation in agile software development’**

A written permission to include this material in its published form [29] has been obtained from Emerald Publishing Limited.

I. A. Tøndel, M. G. Jaatun, D. S. Cruzes and L. Williams, ‘Collaborative security risk estimation in agile software development,’ *Information and Computer Security*, vol. 27, pp. 508–535, 4 2019. DOI: [10.1108/ICS-12-2018-0138](https://doi.org/10.1108/ICS-12-2018-0138) ©Emerald Publishing Limited all rights reserved.



C

# Collaborative security risk estimation in agile software development

Inger Anne Tøndel

*Department of Computer Science, Norges Teknisk-Naturvitenskapelige Universitet, Trondheim, Norway and Stiftelsen for Industriell og Teknisk Forskning, Trondheim, Norway*

Martin Gilje Jaatun and Daniela Soares Cruzes

*Stiftelsen for Industriell og Teknisk Forskning, Trondheim, Norway, and*

Laurie Williams

*North Carolina State University, Raleigh, USA*

508

Received 7 December 2018  
Revised 18 February 2019  
Accepted 27 February 2019

## Abstract

**Purpose** – Today, agile software development teams in general do not adopt security risk-assessment practices in an ongoing manner to prioritize security work. Protection Poker is a collaborative and lightweight software security risk-estimation technique that is particularly suited for agile teams. Motivated by a desire to understand why security risk assessments have not yet gained widespread adoption in agile development, this study aims to assess to what extent the Protection Poker game would be accepted by agile teams and how it can be successfully integrated into the agile practices.

**Design/methodology/approach** – Protection Poker was studied in capstone projects, in teams doing a graduate software security course and in sessions with industry representatives. Data were collected via questionnaires, observations and group interviews.

**Findings** – Results show that Protection Poker has the potential to be adopted by agile teams. Key benefits include good discussions on security and the development project, along with increased knowledge and awareness. Challenges include ensuring efficient use of time and gaining impact on the end product.

**Research limitations/implications** – Using students allowed easy access to subjects and an ability to collect rich data over time, but at the cost of generalizability to professional settings. Results from interactions with professionals supplement the data from students, showing similarities and differences in their opinions on Protection Poker.

**Originality/value** – The paper proposes ways to tackle the main obstacles to the adoption of the Protection Poker technique, as identified in this study.

**Keywords** Case study, Risk assessments, Agile development, Protection Poker, Secure software engineering, Software security

**Paper type** Research paper



This work was supported by the SoS-Agile – Science of Security in Agile Software Development project, funded by the Research Council of Norway (grant number 247678). Thanks to the course organizers of TDT4290 (Prof Jon Atle Gulla and Prof John Krogstie) and the participating students at NTNU and North Carolina State University. Thanks to Tosin Daniel Oyetoyan for contribution to the capstone study. Thanks to Prof Pekka Abrahamsson for input on the capstone study design. Thanks also to the companies that participated in the events and to those helping with facilitation at the security conference (Per Håkon Meland and Marie Moe).

## 1. Introduction

Agile development methods have gained widespread adoption in the software industry, and agile methods are now used for all types of software development and for various types of systems, including large development projects (Dingsøyr *et al.*, 2018). Current evidence shows that security work is often neglected in agile projects (Oueslati *et al.*, 2015; Terpstra *et al.*, 2017; Khaim *et al.*, 2016; Tøndel *et al.*, 2017), and that teams generally do not estimate security risks in an ongoing manner to inform the security requirements work (Tøndel *et al.*, 2017). Risk management is important for making decisions on security activities in agile development, as full security analysis in every sprint is not possible (Oueslati *et al.*, 2015).

Risk-management activities, both within cyber security and software security, are motivated by similar goals: to ensure that security activities are in line with organizational goals and objectives and to address the security needs in an effective and timely manner (ISO/IEC, 2011, Caralli *et al.*, 2007; NIST, 2010). For software security, risk management involves effectively and cost-efficiently identifying security vulnerabilities and risks and prioritizing mitigations (Oueslati *et al.*, 2015) and using risk as a basis for prioritizing development efforts and making trade-off decisions (McGraw, 2006; Chandra, 2008). Additionally, through raising awareness, teams gain an improved understanding of what factors may lead to negative outcomes (Chandra, 2008) and become more able to think like an attacker (McGraw *et al.*, 2016). Achieving these goals is, however, not a straight-forward task. Understanding and assessing security risk is known to be a complex challenge, requiring a large skill set (Tøndel *et al.*, 2017).

Limited empirical data is available on what makes risk management difficult, both within cyber security and software security. A review of risk analysis methods for IT systems (Sulaman *et al.*, 2013) identified a lack of evaluation of risk analysis methods. In spite of the mantra that all security work should be risk based, a study among information security professionals (Jourdan *et al.*, 2010) unveiled that as many as 25 per cent stated that risk analysis was never or rarely performed for their department or organization. A main challenge is the estimation of likelihood and cost, in part because of limited availability of historical data and constantly changing risk factors (Fenz and Ekelhart, 2010; Cybenko, 2006; Gerber and Von Solms, 2005; Rhee *et al.*, 2012; Tøndel *et al.*, 2015).

Few research papers report on risk-analysis methods specifically tailored towards agile teams. Protection Poker (Williams *et al.*, 2010) is a notable exception. Protection Poker is based on the Planning Poker game (Grenning, 2002) that is used for effort estimation in agile projects. Protection Poker is intended to be played as part of every iteration planning meeting to rank the security risk of each feature to be implemented in that iteration and to identify security mechanisms that should be implemented to maintain an acceptable risk level. The full team together identifies assets related to the features and uses the Protection Poker game to rank the features according to their security risk, assessing the value of their assets and the ease of attack. Although proposed by Williams *et al.* (2010) in 2010, Protection Poker is still not widely used in the software industry.

Protection Poker is a promising technique to study further, given its potential to increase security awareness and knowledge in the full development team (Williams *et al.*, 2010). Additionally, previous studies have identified security benefits that can be traced back to using an incremental risk analysis approach (Baca *et al.*, 2015a) and have identified the need for more research on practical ways of doing risk analysis in an agile context, i.e. lightweight and continuously throughout the project (Tøndel *et al.*, 2017). By studying adoption of Protection Poker and how Protection Poker can be successfully integrated into the practices of agile development teams, our aim is to build knowledge on how to increase adoption of security risk assessment practices by agile teams. As Protection Poker has not yet gained widespread

adoption, understanding potential reasons why this is the case can additionally help improve Protection Poker and other techniques with similar goals as Protection Poker.

This paper presents a family of studies of applying Protection Poker in three different settings: by six capstone development project teams; by 16 teams in a graduate software security course; and in sessions with industry practitioners. Our investigation was centred on the following research questions:

- RQ1.* To what extent is Protection Poker accepted by the players, both in the short term and in the longer term?
- RQ2.* What lessons learned and improvements to Protection Poker are identified by the players?

This paper is an extended version of [Tøndel et al.'s \(2018\)](#), which presented the results of using Protection Poker in the capstone development projects. Compared to the previous paper, this paper adds results from sessions of applying Protection Poker with industry representatives and in a graduate-level software security course. Additionally, this paper provides a more thorough overview of literature on security risk assessment in agile development and on challenges to adoption of security activities in agile development. Furthermore, this paper offers more concrete advice on when and how to adopt Protection Poker in agile development projects.

The remainder of this paper is organized as follows. Section 2 gives an overview of relevant literature on adoption of risk-assessment practices in agile software development. Section 3 gives a more thorough introduction to Protection Poker. Section 4 explains the research method used in the study. Section 5 presents the results of the study, and Section 6 discusses the implications of these results. Section 7 discusses threats to validity. Section 8 concludes the paper.

## 2. Adoption of security risk-assessment practices by agile teams

This section gives an overview of the current state of software security risk assessment in agile software development and introduces literature on challenges and factors important for adoption of security activities in agile development.

### 2.1 Approaches to security risk assessment in agile development

Software development projects need to deal with various types of risks, including security risks. In agile development practices, risk management can be said to be treated implicitly ([Tavares et al., 2017](#); [Odzaly et al., 2018](#)), and the guidance provided by agile methods when it comes to risk management is “very general” ([Nyfjord and Kajko-Mattsson, 2008](#)). Few research papers provide concrete guidance on how to tackle security risk management for agile projects ([Tøndel et al., 2017](#)). The most notable technique available is Protection Poker. Additionally, the security-enhanced agile software development process (SEAP) studied at Ericsson ([Baca et al., 2015b](#)) provides high-level suggestions for performing incremental risk analysis, in addition to other practices such as adding more security resources to the teams and performing security activities such as code review and penetration testing. The results of a study of SEAP reported improved identification and handling of risk. Thus, risk management was found to be more cost-efficient with SEAP than with the approach previously used by Ericsson, because security issues were dealt with in a more distributed fashion with more issues solved directly by the team. The details of how risk analysis and risk management was conducted with SEAP is not available; however, the frequency of risk analysis was increased, the scope for each analysis was reduced and the approach was more distributed.

Within the area of cyber security, standards, guidelines and research papers suggest different ways of managing risk and performing risk assessments. Some of the major ones are ISO/IEC 27005 (ISO/IEC, 2011), OCTAVE Allegro (Caralli *et al.*, 2007) and the NIST Risk Management Framework (RMF) (NIST, 2010). These documents suggest practices that concern assessing the risk, making decisions on how to treat the risk, following up on these decisions and communicating information related to security risks in the organization (Tøndel *et al.*, 2017).

Within software security, risk-assessment practices are commonly included in software security frameworks, maturity models and security development lifecycles (SDLs). The OWASP Software Assurance Maturity Model (OpenSAMM) (Deleersnyder *et al.*, 2017) includes activities on performing threat assessments. The seven touchpoints for software security (McGraw, 2004) include the touchpoints abuse cases and risk analysis. Microsoft SDL (Howard and Lipner, 2006) includes activities to perform security and privacy risk assessments, attack surface analysis/reduction and perform threat modelling. One of the main sources for information about software security practices is the Building Security In Maturity Model (BSIMM) (Williams *et al.*, 2018), mainly giving an overview of practices of big companies. Though BSIMM does not have an activity that is named “risk analysis”, it contains several activities related to such an activity, e.g. “Use a risk questionnaire to rank applications”, which has an adoption of 45 per cent in the 2017 version of BSIMM, and “Require security sign-off” which concerns a process for risk acceptance and has an adoption of 30 per cent. Regarding smaller companies, we are only aware of one study in this respect (Tøndel *et al.*, 2017), finding that risk-assessment practices in public development organizations were not based on risk analysis, but rather driven by compliance. The organizations performed risk analysis on some level, but the practices were by and large not integrated with and considered to be especially relevant for development. Thus, more empirical studies are needed on what can be done to increase adoption of security risk assessment by software development projects.

Table I provides an overview of the risk management approaches mentioned above and their usefulness for agile development teams. Note that a complete overview of available methods is outside the scope of this paper.

### 2.2 Challenges of security activities in agile development

Research in software security covers a varied range of approaches and processes that deal with security during software development. Several approaches have been suggested to incorporate security into the software development lifecycle (Oueslati *et al.*, 2015). When aiming to apply security practices in agile software development, it is essential to take into consideration the agile principles of “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). Current studies have identified challenges when security work is expected to be guided by these same principles (see Tables II and III for an overview of the cited studies).

Oueslati *et al.* (2015) identified a set of 14 challenges of developing secure software using the agile development approach and methods reported in the literature. The challenges were categorized as “Software development lifecycle challenges”, “Incremental development challenges”, “Security assurance challenges” and “Awareness and collaboration challenges”. Several of the challenges relate to the need to fit security activities into the short iteration times, the need to deal with changing functional requirements that may break previous security analysis and decisions and the reliance on documentation that is common within security work.

In a case study in a development organization, Cruzes *et al.* (2018) identified challenges to threat modeling in agile development. The principle of “Individuals and interactions over processes and tools” was challenging, especially because it was hard to get effective

ICS  
27,4

512

**Table I.**  
A selection of  
existing approaches  
and their agile  
usefulness

Approach and references	Risk-management activities included	Usefulness for agile teams
<i>Agile methodologies concerning security risks</i>		
SEAP (Baca <i>et al.</i> , 2015b)	High-level suggestions for risk analysis	Details are not available
<i>Risk-analysis methodologies</i>		
ISO/IEC 27005 (ISO/IEC, 2011)	Information security risk management Organizational approach	Not directly applicable to agile software development
OCTAVE Allegro (Caralli <i>et al.</i> , 2007)	Information security risk management Organizational approach	Not directly applicable to agile software development
NIST Risk Management Framework (RMF) (NIST, 2010)	Integration of risk management into software development	Describes a continuous process, but may be too comprehensive for many agile projects
<i>Software security maturity models</i>		
OpenSAMM (Deleersnyder <i>et al.</i> , 2017)	Contains activities for threat assessment, security requirements and more	Activities can be adopted by agile teams High-level descriptions
BSIMM [16] (Williams <i>et al.</i> , 2018)	Based on studies of predominantly large companies Contains some risk-related activities, e.g. attack models	Activities can be adopted by agile teams High-level descriptions
<i>Secure software development lifecycles</i>		
Touchpoints for software security (McGraw, 2004)	Includes abuse cases and risk analysis	Activities can be adopted by agile teams High-level descriptions
Microsoft SDL (Howard and Lipner, 2006)	Includes security and privacy risk assessments, attack surface analysis/reduction, threat modelling	Agile version of this SDL is available (Microsoft, 2012) that explains how to integrate these practices in agile development

meetings with clear and actionable outputs. The principle of “Working software over comprehensive documentation” is many a time misunderstood by agile teams to be “no documentation” and especially developers have lost the focus on documenting their work or understanding the need for documentation. Security work is many a time based on the documentation of the decisions, risks and assets. This study gives us motivation to investigate further how to make the security discussion meetings more effective.

Türpe and Poller (2017) theorize about tensions between the characteristics of security requirements and security work on the one hand and the way Scrum manages development work on the other. The authors find three different ways of managing security work: as bug fixing on demand, continuously as a quality requirement through the definition of “done” or as prioritized and planned development work through the product backlog. All of them are found inadequate. On-demand fixing rarely leads to substantial security improvement. As a quality requirement, security has a complex relationship with development work and is difficult to verify. Security features in the backlog would be a suitable approach to many security concerns, but they compete with other requirements and may also need special expertise to design and implement effectively.

Terpstra *et al.* (2017) conducted a study of how practitioners reason about and cope with security requirements in agile development, based on postings on LinkedIn. They identified 21 challenges and 15 coping strategies, and used these to create a conceptual model. Challenges pointed to in this model is a limited business case for security, unclear ownership of security requirements, limited organisational effort to educate developers on security,

Author and reference	Topic	Challenges
Oueslati <i>et al.</i> (2015)	Software security in agile development (literature review)	<p>“Software development life-cycle challenges” (security activities not included; hard to integrate security in every iteration because of short iteration times)</p> <p>“Incremental development challenges” (dealing with changes)</p> <p>“Security assurance challenges” (documentation; testing; unstable development process)</p> <p>“Awareness and collaboration challenges” (security requirements neglected; lack of experience and security awareness; separate the developer and reviewer roles)</p> <p>“Security management challenges” (giving priority to security)</p>
Cruzes <i>et al.</i> (2018)	Threat modelling in agile (case study)	<p>“Asset Identification” (documentation)</p> <p>“Data Flow Diagrams” (documentation; level of abstraction; interfaces; link to code; maintainability)</p> <p>“Modeling Meeting” (effective meetings; who should participate; distributed settings; finding what is good enough; expertise)</p> <p>“STRIDE” (communication channel focus)</p> <p>“Outputs from the Session” (make it actionable; follow up; prioritizing security; false sense of security)</p>
Türpe and Poller (2017)	Security requirements in SCRUM	Security as bug fixing on demand, continuously as a quality requirement, or as prioritized and planned development work through the backlog
Terpstra <i>et al.</i> (2017)	Security requirements in agile (study of practitioners’ posts on LinkedIn)	<p>Unclear business case for security (hard to sell as business value; costly)</p> <p>Unclear ownership of security requirements (forget about security; delivered late)</p> <p>Perceptions of priority (differing priorities; not prioritized by customers and product owners)</p> <p>Understanding of security (low awareness among developers; lack of training; dependent on individuals)</p> <p>Organizational context (lack of involvement of security experts; product owner becomes a limiting factor; organizational structure can make or break modifications to requirements)</p> <p>Poorly defined security requirements</p>
Alsaqaf <i>et al.</i> (2017)	Quality requirements in agile (literature review)	<p>Technique (no widely accepted technique; inadequacy of existing techniques; traceability)</p> <p>Priorities (functionality is prioritized; ignore some types of requirements; validated late; insufficient analysis)</p> <p>Product owner (lack of knowledge; workload; availability; dependence)</p>

**Table II.**  
Selected challenges identified in the literature

limited incentives to care about security and the varying perceptions of priority among business representatives. The agile principle of “Individuals and interactions over processes and tools” lead to the priorities of security work being highly reliant on the people involved in the project. Both [Alsaqaf \*et al.\* \(2017\)](#) in a literature review of quality requirements work in agile development, and [Terpstra \*et al.\* \(2017\)](#) identify the product owner as a hindrance for quality requirements being properly addressed, as the product owner commonly have a “heavy workload” and “insufficient availability”, in addition to a “lack of knowledge” on the

ICS  
27,4

514

**Table III.**  
Selected factors  
identified in  
literature, influencing  
adoption of software  
security practices

Author and reference	Topic	Factors
Kanniah and Mahrin (2016)	Implementation of secure software development practices (literature review)	<p><i>“Institutional Context”</i> (change management; policy enforcement; training; incentives; culture and organizational objectives; security experts)</p> <p><i>“People and Action”</i> (developer and project manager attitude and skills; management support)</p> <p><i>“Project Content”</i> (tool support; budget; security experts/team; development time; development methodology)</p> <p><i>“System Development Process”</i> (methodology; security requirements; security policies/standards/guidelines; metrics/KPI)</p>
Geer (2010)	Adoption of secure SDLs (online survey)	<p>The three most popular responses were</p> <p><i>“Too time-consuming”</i></p> <p><i>“Not aware of methodologies”</i></p> <p><i>“Requires too many resources”</i></p>

quality aspects (Alsaqaf *et al.*, 2017). Challenges on prioritizing time and money on security are however not specific for agile development. Kanniah and Mahrin (2016) has previously in a review of 44 primary studies identified “Adequate Development Time” and “Adequate Budget/Cost” as commonly cited factors that impact the successful implementation of secure software development practices. Additionally, Geer (2010) found in a survey that “Too time-consuming” and “Requires too many resources” were the main self-reported reasons for not adopting secure SDLs, together with not being aware of the methodologies.

### 3. Protection Poker

Protection Poker is a technique that is designed to engage the whole team in discussing security problems, and does so in a way that is concrete and (hopefully) fun. Additionally, Protection Poker is specifically designed to be applied for each development iteration, ensuring that security is considered throughout. Protection Poker results in a ranking of features based on their security risk. Protection Poker was originally proposed by Williams *et al.* (Williams *et al.*, 2010; Williams *et al.*, 2009), and later modified by Jaatun and Tøndel (2016). As can be seen from Table IV, these two variations of Protection Poker differ on two aspects: the risk calculation and the card values used.

Protection Poker is designed to be played during an iteration planning meeting with the participation of the full development team. One person should have the role as moderator, and this person will be responsible for leading the team through the game and pointing the discussions in a good direction. Ideally, a separate person should be tasked with taking notes on important security solutions and ideas that emerge during play. Focus is on the specific requirements the team will likely implement during the next iteration. A basic

**Table IV.**  
Overview of the two  
Protection Poker  
variations

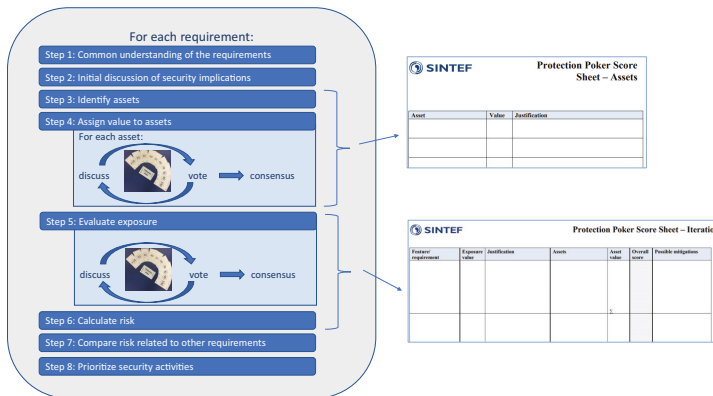
Aspect	Original Protection Poker (Williams <i>et al.</i> , 2010)	Modified Protection Poker (Jaatun and Tøndel, 2016)
Risk calculation	$\text{risk} = \sum(\text{asset values}) \times (\text{ease of attack})$	$\text{risk} = \sum(\text{asset values}) \times (\text{exposure})$
Card values	1, 2, 3, 5, 8, 13, 20, 40, 100	< 10, 20, 30, 40, 50, 60, 70, 80, 90, 100
Used in study	Graduate projects	Capstone projects, industry events



overview of the steps involved in playing Protection Poker can be found in Figure 1. The actual playing using the Protection Poker cards is done in steps 4 and 5. Players use the cards to make votes on the risk involved in the requirement they are playing on, and the votes are a basis for further discussions on the risk and eventually agreeing on a risk value for the requirement. This agreement may require several rounds of voting by using the Protection Poker cards. Below we explain two central concepts of the game, namely, risk calculation and calibration.

Risk is always related to a requirement that is to be implemented in the next iteration, which is often new, enhanced or corrected functionality. Exposure/ease of attack relates to how hard or easy the added or changed functionality makes it to attack the system. For asset value, one identifies the assets that are related to a requirement and considers their value for various actor types. Assets are typically considered to be “data stored in database tables or system processes that the new functionality controls” (Williams *et al.*, 2010); however, in this study we did not use a strict definition of the term asset. In previous work (Jaatun and Tøndel, 2008), we have defined assets as “anything of value that needs to be protected”.

To be able to prioritize between requirements and to avoid that high-risk projects assign every requirement a high risk value, the numbers assigned need be spread. Thus, the highest card (100) should be used for asset values and exposures/ease of attack that are high for this project, and similarly the lowest card (< 10 or 1) should be given to asset values and exposures/ease of attack that are low for this project. The goal is not to establish a “perfect” and “universal” risk value but rather to rank the security risk of the requirements to be able to better prioritize security effort. Therefore, a calibration is recommended in the beginning of the playing Protection Poker to arrive at a common understanding of the end-points of the scale, i.e. the team agrees what a ‘< 10/1’ or a “100” means for this product. When playing about asset value and exposure/ease of attack, numbers should be assigned relative to these endpoints, as well as relative to the values assigned for previously assessed assets and features.



Source: Tøndel *et al.* (2018)

Figure 1.  
Playing Protection  
Poker (modified  
version)

#### 4. Research methodology

This section gives an overview of the research method used for this study. The study was performed in three parts:

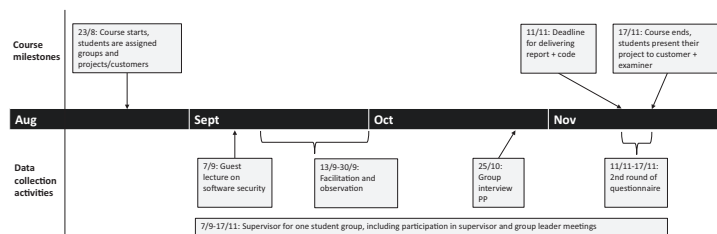
- (1) a case study of six capstone projects lasting about three months, all using the modified Protection Poker version;
- (2) single sessions of playing Protection Poker with industry practitioners, either at company visits or at conferences, all using the modified Protection Poker version; and
- (3) a case study of 16 project teams in a graduate-level software security class, using the original Protection Poker version.

##### 4.1 Capstone projects

The study was performed in the Customer Driven Project course (TDT4290) at the Norwegian University for Science and Technology (NTNU), Autumn 2016. This course is mandatory for fourth-year computer science students. In this course, the students are divided into development teams (five-eight students per team). Every team is given a development project from an external customer (i.e. private companies, public organizations or research institutes). The students are expected to investigate the needs of the customer, develop software, do some testing of this software and document everything in a report and in a presentation given to the customer. In general, all student groups use agile methodologies to some extent. Six groups, consisting of 34 students in total, were required to use Protection Poker for their project. These groups developed various systems: an app for pupils, a game, an algorithm, a Web-based system to register projects and an isolated system for annotating safety arguments (Figure 2).

An overview of data-collection activities can be found in Figure 1. As most students had received limited formal training on software security before this course, we arranged a lecture where all students were given a short plenary introduction to software security and the Protection Poker game. They played the game on an example project and responded to a questionnaire that covered the students' acceptance of the technique. Data collection proceeded through facilitation and observations of students playing Protection Poker in their group, and the observations were followed by group interviews towards the end of the course, allowing detailed student feedback on the technique. Additionally, the main author of this paper acted as supervisor for one of the student groups and took part in project manager and supervisor meetings throughout the course. The questionnaire on acceptance was repeated towards the end of the course. The study has been reported to the national Data Protection Official for research.

**Figure 2.**  
Overview of data-  
collection activities



Source: Tøndel *et al.* (2018)

The main motivation for using a questionnaire was to capture students' immediate and longer-term acceptance of the Protection Poker technique (*RQT*). A questionnaire could easily reach many students and could easily be repeated. We decided to base the questionnaire on the technology acceptance model (TAM) (Davis, 1985) for two reasons. First, TAM, although criticized by some (Li, 2010), is considered a highly influential and commonly employed theory for describing an individual's acceptance of information systems. Thus, we believed TAM could help us understand the different reasons for acceptance of Protection Poker by the students, and that TAM-based questions could trigger comments from the students related to acceptance. Second, we were able to adapt questions from an existing questionnaire (Caroli and Caetano, 2015) to the phenomena we are studying.

The TAM, adapted from the theory of reasoned action (Ajzen and Fishbein, 1980) and originally proposed by Davis, suggests that when users are presented with a new technology, a number of factors influence their decision about how and when they will use it, notably:

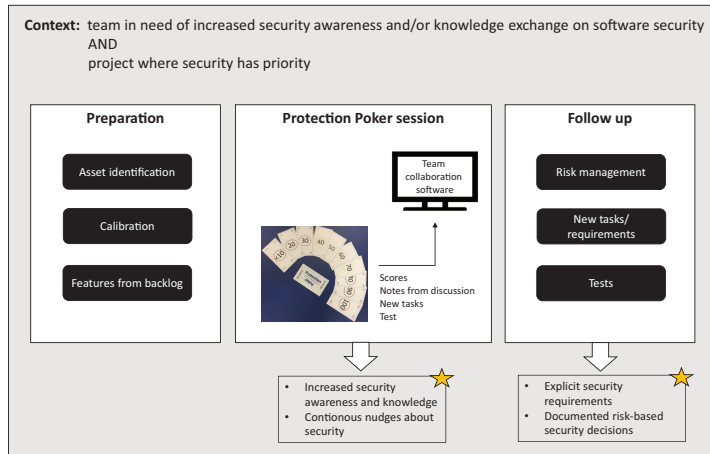
- *Perceived usefulness*: This was defined by Davis as “the degree to which a person believes that using a particular system would enhance his or her job performance” (Davis, 1989).
- *Perceived ease of use*: Davis defined this as “the degree to which a person believes that using a particular system would be free from effort” (Davis, 1989).
- *External variables*: These include “system characteristics, training, user involvement in design, and the nature of the implementation process” (Venkatesh and Davis, 1996).

For the observations, we created a rota where one of the authors served as facilitator and at least one other author participated as observer. After each observation session, both the facilitator and the observer filled in reflection notes in a template that contained the following topics: group information; questions from the students on the technique; suggested changes to the game; participation; mood; topics discussed; what worked well with the game; challenges with the game, and; reflections on the observation; and how the researchers may have influenced the process. After playing one session of Protection Poker, all groups were encouraged to keep on playing by themselves during the project, and we offered to return and offer support and/or facilitation at a later time, according to their needs.

When we facilitated the students in the capstone projects in playing the Protection Poker, we covered steps 1-6 in Figure 3, in addition to calibration. Each Protection Poker sessions lasted between 50 and 70 min and contained the following activities:

- *Introduction*: The session started by having the students explain their system to the facilitator.
- *Assets*: We prioritized calibrating the top end of the scale. The groups played on two-three assets and spent between 1 and 17 min per asset played. For most (10 of 14) of the assets, the students were able to agree on a value with two rounds of playing the cards.
- *Features*: Calibration of features was skipped in three of the groups because of limited time left. We prioritized playing about features over identifying and calibrating features. One group did not play on any features, because the nature of their project (creation of an algorithm) made it difficult to come up with features. The other five groups played on one to three features. The students spent between

**Figure 3.**  
Overview of our  
suggested approach  
to adopting  
Protection Poker



2 and 9 min per feature played. For all but one feature, two rounds of play were necessary.

- *Reflection:* The session ended with reflection about the experience, and the students were asked to provide feedback and suggest improvements.

Throughout, the facilitator was active in helping the group reach a consensus by suggesting compromise values. This facilitation was done to speed up the playing, terminating discussions when most arguments had been raised.

Towards the end of the course, all groups were invited to send two-three participants to an event where the technique would be discussed in more detail. This event was organized as a group interview and was scheduled to last for 2 h. The following topics were covered: students' expectations to the event; use of the game in the group; brainstorming and discussion on the 4Ls (Liked, Lacked, Learned, Longed for) (Caroli and Caetano, 2015); suggestions for improvements to the technique; suggestions for improvements to how software security was handled in the course, and; feedback on the event. Discussions were recorded and transcribed. To encourage participation, all participants were served pizza and they had the opportunity to win cinema gift cards. Non-responding groups were reminded via email. To promote active participation in the group interviews, each event was split in two parallel sessions. Note that in the observations, we found that only two of the six groups had obvious security concerns. The group interview had low participation from those groups; only one participant from only one of those groups, while all the groups with limited security concerns participated with two-three people.

#### 4.2 Industry practitioners

We have introduced Protection Poker to several of our industry collaborators, and in some cases we have been able to observe gameplay and collect questionnaires afterwards. This paper covers four interactions in companies and one interaction at a security conference aimed at industry. An overview of these interactions is given in Table V. The questionnaires used for data collection were a variation of the questionnaire used for the students, with the same TAM-based questions. The sessions (events 1-4) were organized in the following way:

#	Where	Who	What	When	Data collected	No. of participants
1	Software development company	Developers, architects, security officer	Played PP on items in their backlog	June 2016 (2h)	Questionnaire responses (4), observation notes (2)	14 (3 groups), in addition to participants listening in from another site
2	Security conference for industry	Product owner (1), manager (1), security architect (1), other security roles (10)	Played PP on synthetic case	September 2016	Questionnaire responses (13), notes from discussion with two facilitators	20
3	Software development company	Project manager and developers	Played PP on items in their backlog	October 2017	Questionnaire responses (5)	5
4	Software development company	Developers, security manager, architects	Played PP on items in their backlog	January 2017 (1,5 h)	Observation notes (1)	8 on site, 1 listening in (one group)
5	Software development company	Security officer	Presented PP and got feedback on suitability for their organization	June 2016	Notes (1)	1

**Table V.**  
Overview of events with industry practitioners

ICS  
27,4

520

- *Presentation of Protection Poker*: The events all started with a presentation essentially identical to the one given to the capstone student groups.
- *Playing in groups, with facilitator*: The groups played a few rounds, either on a synthetic case (event 2) or on features in their own backlog (event 1, 3 and 4). Note that in event 1 there were only two facilitators, so the other groups (one local and some remote) had to manage without support.
- *Reflections and questionnaire (in events 2 and 3)*: In event 1, the participants did not have time to fill out the questionnaire in the session but were asked to do this after the session and deliver to the security officer.

#### 4.3 Graduate-level software security course

Protection Poker was also studied in the graduate-level Software Security course (CSC515) in the computer science department at the North Carolina State University during the Autumn 2018 semester[1]. This course is an optional elective for master and PhD students. The purpose of the course is to introduce students to the discipline of designing, developing and testing secure and dependable software-based systems. During the 15th week of the semester, the students were given an approximately 30-min lecture on the Protection Poker technique. After the lecture and during the same class period, the students did a 30-min in-class exercise using the technique on a sample set of requirements.

In this course, the students are divided into 16 project teams (2-3 students per team, 58 students total). Through five deliverables during the 15-week semester, every team analysed the security of the OpenMRS electronic health record application using a variety of techniques and tools[2]. As part of the fourth deliverable, the students were asked to write five new functional requirements for Open MRS to add functionality that is not in the system yet. Each team then played Protection Poker on these requirements. This activity was not conducted in the presence of any of the teaching staff. After the deliverable was turned in, the students were asked to complete a questionnaire similar to that which was administered to the capstone groups. The students completed the questionnaire during class. The students were informed that the questionnaire response was part of a study and that their participation was optional; 46 students completed the survey. This study was approved by the university Institutional Review Board.

#### 4.4 Analysis

All questionnaire data were analysed using descriptive statistics, to get an overview of responses to the individual questions. We did not undertake further analysis of this data, as the number of responses from each part of the study were limited.

The qualitative data from the capstone development projects (observations, group interviews) were coded and organized within the Mind Manager tool. The qualitative data came in different forms, and the coding and organising of the data was first done for one type of data at the time. As an example, the template for observation notes contained a table to note what worked well and what was challenging with specific aspects of Protection Poker (Tøndel *et al.*, 2018). All the data collected in that table was analysed together. In a similar way, all the results from the 4L exercise in the group interviews were analysed together (Tøndel *et al.*, 2018) to identify what were the common responses. Then, the transcribed group interviews and the free text observation notes were coded together with the more structured qualitative data in the same mind map to extend the findings and to identify other topics. These activities resulted in the identification of main areas where

improvement was needed (Tøndel *et al.*, 2018) and the benefits and challenges of the Protection Poker, taking only results from the capstone development projects into account.

The qualitative data from the interactions with industry representatives and from the study at the graduate security course were less extensive. The reflection notes from some of the industry events, and the responses given to the open-ended question in the questionnaire, were coded to see if there were data that supported or conflicted with the benefits and challenges identified in the capstone projects. In addition, we looked for challenges and benefits not already identified in the capstone projects.

## 5. Results

This section presents the results according to the two research questions of this study. Compared to Tøndel *et al.* (2018), this section gives an overview of results from industry events and a graduate course, in addition to the capstone projects. However, this section does not give as detailed an overview of the data collected from the capstone projects (observations, questionnaire results, 4 L brainstorming and quotes from interviews) as the previous version. Readers who want more details on the results from the capstone project study are therefore referred to Tøndel *et al.* (2018).

The results point to one factor apart from Protection Poker that may have had a major impact on the results, namely, the limited need for security in four out of the six capstone projects. The results relating to this factor are thus given special attention. An overview of the benefits and challenges identified with Protection Poker in this study can be found in Tables VI and VII.

### 5.1 Acceptance of Protection Poker (RQ1)

Acceptance of Protection Poker (RQ1) was mainly studied through the TAM-based questionnaire. Table VIII gives a high-level overview of the responses to the questionnaire. Detailed results can be found in Tøndel (2018). The responses come from different sources:

- responses from the students of the capstone project after the introductory lecture (29 responses);
- responses from the students in the capstone projects at the end of the course (30 responses);
- responses from industry representatives taking part in playing Protection Poker at different events (21 responses); and
- responses from students in a graduate-level course after using Protection Poker to analyse the security risk of new requirements for their semester-long project (46 responses).

Four questions together cover the variable *future use intention* (“I intend to increase my use of the PP for project-work in the future”; “I intend to use the PP in the future for my projects”; “Given a choice, I would prefer not to use the PP in any future projects”; “I would like to use the PP in the future”). In the capstone projects, responses show that the students tend towards being positive to use Protection Poker. In the end, half (15) of the capstone project students agree that they would like to use Protection Poker in the future, while only 5 did not want to use Protection Poker. Among the industry representatives the responses are quite similar, with 12 wanting to use Protection Poker, while only 2 did not want to use Protection Poker. The graduate students in the software security class were the most positive to using Protection Poker, where 38 agreed that they want to use Protection Poker, while only 3 disagreed.

Benefit	Capstone projects	Industry	Graduate software security
Playing PP is perceived as useful (B1)	Learned things from playing PP. Helped them think about security. Gave overview of project (questionnaire, group interviews)	Expectation that PP can improve security (questionnaire)	<i>(Strong evidence)</i> PP is found to improve security (questionnaire)
PP is easy to learn (B2)	<i>(Strong evidence)</i> Most students agree that PP is easy to learn (questionnaire)	<i>(Limited evidence)</i> Respondents are neutral on this aspect in the questionnaire responses	Most agree that PP is easy to learn (questionnaire)
PP is easy to use (B3)	Most find PP easy to use (questionnaire) Observations and group interviews identify challenging aspects	<i>(Limited evidence)</i> Respondents are neutral on this aspect in the questionnaire responses	Most agree that PP is easy to use (questionnaire)
PP brings about useful discussions in the team (B4)	<i>(Strong evidence)</i> View expressed in group interviews and supported by observation	Supported by observation	Discussions bring a deeper understanding and help reveal security issues (response to open-ended question)
PP makes everybody participate in security discussions (B5)	Supported by observations and group interviews. However, we observed that some are passive	Observed that, although there was a homogenous group, they still ended up with different scores in the beginning. Observed in one company that team members did contribute to the discussion although they initially believed they did not know much	Contributions from team-members with different perspectives are useful (response to open-ended question)
The relative scale makes PP useful also for projects where security is not a major issue (B6)	View expressed in group interviews	<i>(Not studied)</i>	<i>(No evidence)</i>
Results from playing PP are easy to interpret, and can be used for prioritization (B7)	View expressed in group interviews. Especially they liked the overview of the assets	Liked the way PP estimate risk (response from one company representative)	PP can lead to risk reduction by modifying requirements (response to open-ended question)
PP is fun to play (B8)	View expressed in group interviews	<i>(No evidence)</i>	<i>(No evidence)</i>
Increase knowledge and awareness about security (B9)	View expressed in group interviews	Observed discussions that increased security knowledge among participants (example: A: "to attack would you not need to . . ." B: "but that is not so difficult because. . .")	<i>(No evidence)</i>

**Table VI.** Overview of benefits identified, with an indication of where evidence is particularly strong and where there may be limited evidence or no evidence



Challenge	Capstone projects	Industry	Graduate software security
PP poker did not improve security of the software (C1)	<i>(Strong evidence)</i> Results from PP does not directly influence development (questionnaire, group interviews)	<i>(Not studied)</i>	<i>(Conflicting evidence)</i> Improved security is considered a main benefit from PP (questionnaire) and students express that playing PP has made an impact through modifying requirements (response to open-ended question)
Limited relevance for their project (C2)	Four of six projects had few security concerns	<i>(Not studied)</i>	<i>(Not studied)</i> The OpenMRS had many security concerns
Starting to use PP is time-consuming because of calibration and the need to identify and play about assets (C3)	<i>(Strong evidence)</i> Supported by observations. The time needed to play was considered a challenge in group interviews	Observed that calibration and asset identification for the first feature took a lot of time	
It is difficult to reach consensus, something that results in a lot of time spent and sometimes result in tension in the team (C4)	View expressed in group interviews and supported by observations. But in some of the group there were no problems related to this (observations)	<i>(No evidence)</i>	One student expressed challenges related to conflicts (response to open-ended question)
Some team members may end up with too much influence (C5)	View expressed in group interviews and supported by observations	<i>(No evidence)</i>	<i>(No evidence)</i>
Ensuring confidence in the result (C6)	View expressed in group interviews	<i>(No evidence)</i>	<i>(No evidence)</i>
The relative scale can be difficult to understand (C7)	Observed in one group. View expressed in group interview	<i>(No evidence)</i>	<i>(No evidence)</i>
Selecting granularity of assets and assigning value to assets can be challenging (C8)	View expressed in group interview	Some expressed that assets should be identified beforehand to ensure they are at a common level	<i>(No evidence)</i>
The term exposure is difficult to understand (C9)	Observed in several groups	Observed in one of the companies	<i>"It's hard to know what is easy to access and what is not"</i> (response to open-ended question)
Exposure and asset value are often mixed up in the discussions (C10)	<i>(Strong evidence)</i> Many questions on the terms, and the terms were often mixed up in discussions (observations)	Observed in at least two events. Possible misunderstanding; exposure for each asset	<i>(No evidence)</i>

*(continued)*

**Table VII.**  
Overview of challenges identified, with an indication of where evidence is particularly strong and where there may be limited evidence or no evidence

Challenge	Capstone projects	Industry	Graduate software security
Important aspects from the discussion is lost (C11)	Observed in several groups	In one company it was suggested to improve this by having something on Jira	(No evidence)
Teams did not end up using PP in a regular fashion (C12)	(Strong evidence) Only one of the student groups used PP on their own, and then only once	(Not studied)	(Not studied)
Planning meetings are already full (C13)	(Not studied)	Response from discussing Protection Poker with company representatives	(Not studied)
Scalability across teams (C14)	(Not studied)	In one company it was pointed out that features and tasks could be moved across teams, and that they had a common backlog. Then it would be easier if all teams used the same assets and had the same asset values for them	(Not studied)
Many cards (C15)	Response from some students in group interviews	Response from some players in one of the companies	(No evidence.)
The output from playing PP is not concrete in terms of what to do next (C16)	Lacked discussions on how an attack could happen (group interviews)	In the observation notes from one company it was noted the lack of a “practical product” from the meeting	(No evidence)
PP takes too much time (C17)	Observed when beginning to use PP, but do not know how this will be later. Questionnaire responses slightly disagree. See also C4	Neutral questionnaire responses. But concerns that teams will not be able to use the technique (ref. C13)	One student stated that PP was “cumbersome and time consuming”, another that it was more formal than what would be expected in companies (response to open-ended question). Neutral questionnaire responses

Table VII.

Four questions together cover the variable *perceived usefulness* (“I think PP will be useful in my current project”; “Using the PP will improve the security of the product”; “Using the PP will substantially reduce the number of serious security defects”; “The advantages of using the PP outweighs the disadvantages”). Many students in the capstone projects found Protection Poker to be useful; in the end more students agreed (14) than disagreed (4) that the advantages of using Protection Poker outweigh the disadvantages. Expectations among the capstone project students on what Protection Poker would deliver was in general high, however, it seems that Protection Poker did not quite deliver in their current project. In

particular, Protection Poker does not seem to have delivered on security – not improving the security of the product and not reducing security defects. The responses from the industry representatives are very much in line with the responses from the students before they had used Protection Poker in their own project, with one exception: the industry representatives are more positive regarding the results from playing Protection Poker on the security of the product. On the question “*Using the PP will improve the security of the product*”, 12 of the 21 respondents agreed, and 4 strongly agreed, while only 1 disagreed. The graduate students in the software security class were even more positive on the usefulness of Protection Poker, especially on its ability to improve security; 25 students agreed, 12 strongly agreed and none disagreed.

Six questions together cover the variable *perceived ease of use* (“Learning to use the PP was easy for me”; “I think the PP is clear and understandable”; “Using the PP requires a lot of mental effort”; “I find the PP easy to use”; “The PP is cumbersome to use”; “Using the PP takes too much time from my normal duties”). Overall, the capstone project students’ responses to these questions were positive, and increasingly so towards the end of the course. To illustrate, in the end only one of the capstone project students found Protection Poker to be difficult to learn, as opposed to 25, who found it easy, and the majority of the students ended up finding Protection Poker to be clear and understandable (22 students) and easy to use (23 students). The responses from the industry representatives however show that they did not find Protection Poker to be as easy to use, and their responses were overall neutral on the corresponding questions. The graduate students in the software security class responses generally lie between these two groups.

### 5.2 Lessons learned and improvements identified by the players (RQ2)

Lessons learned and improvements (RQ2) were studied through observations and group interviews in the study of the capstone projects. Two main areas were identified where improvements were needed; the discussions and the scores and scales used. In the following, we introduce these areas and the improvements suggested by the students related to these areas.

The discussions resulting from playing Protection Poker were considered highly useful by many of the students that participated in the group interviews, but at the same time they reported on several challenges related to keeping the discussions effective and efficient. Key concerns by the students were that:

- Some players end up with too much influence because of their personality.
- Difficulties in reaching consensus results in fighting instead of a common understanding.
- Protection Poker sessions take time.

These benefits and challenges were supported by observation notes from the researchers as well. In the observation notes, half the capstone groups (3) were characterized either by dominant or passive participants, something that negatively influenced the general mood

TAM variable	Capstone projects – <i>before</i>	Capstone projects – <i>after</i>	Industry representatives	Graduate software security
Future use intention	Somewhat agree	Somewhat agree, but less so	Somewhat agree	Agree
Perceived usefulness	Somewhat agree	Neutral	Somewhat agree	Agree
Perceived ease of use	Somewhat agree	Agree	Neutral	Somewhat agree

**Table VIII.**  
High-level overview  
of responses to  
questionnaires

while playing. In all groups, the facilitator was quite active in supporting the students in reaching a consensus. An additional challenge observed was that important aspects from the discussions got lost, as it was not noted down anywhere.

Although students did not have any clear suggestions for improvements that directly address the challenge of having both good discussions and efficient playing of Protection Poker, they had suggestions that could partly help improve the challenges related to the discussions. One suggestion was to have fewer cards, and thus a more coarse-grained scale. Another suggestion was to have more support on security in form of what to discuss and how to ensure they were on the right track. Both the response from the students after the sessions and our own observations suggest that it would have been very difficult for the students to start using Protection Poker without an external facilitator that could help on the game and bring in software security competence. Though the need for an external facilitator was clearly expressed by the students, it is important to add that one group played Protection Poker on their own after the supported session, and they reported that this had gone very well, and in some ways better because they did not have to explain the system to someone external.

Challenges relating to scales and scores concerned two main issues: understanding the relative scale, and understanding the concepts *asset value* and *exposure*. Having a relative scale was considered a benefit by some students, as it made Protection Poker a useful technique also for projects without any major security issues. However, the scale was considered difficult to understand and relate to by other students; You did not know if a “100” was Armageddon or it was just “we need to look into this”. Additionally, disagreements on how to understand the scale slowed down playing in some groups. Some of these challenges that we experienced with the scale may be related to us skipping calibration of the low end of the scale to save time. The improvements suggested by the students related to the scale go in two directions: to explain the relativity of the scale better, or to change the scale. The latter suggestion was less common. Additionally, students suggested to take the time to do a full calibration.

The terms *asset* and *exposure* seemed to be new to many of the students, and in the observations the students had many questions on these terms. The terms were often mixed up in the discussions, with students talking about the exposure of an asset or the value of a feature. Especially the term exposure was found difficult to describe in a good way. For assets it was sometimes difficult to know how to assess their value, as the value may be different if you consider just confidentiality than if you include other aspects of its value as well. Another challenge identified by the students was how to divide up assets in a way that is consistent and does not impact the scores in an unintended way. They pointed out that if you have assets at different levels of granularity this may skew the scores; a feature with many assets of low granularity may get a higher score, and thus priority, than a feature that has assets with higher granularity. Note however that in spite of these challenges, students expressed that they found the end result to be easy to interpret, that is was predictable because of the process and that it gave them a nice way to prioritize the assets of the project. Students did not suggest any changes to these terms, but they suggested to identify assets in advance, and provide better guidance on how to identify assets.

### 5.3 Effect of limited security issues in the capstone projects

The capstone project study found that students perceived little benefits from playing when it comes to security. The open-ended responses on the questionnaire shed some light on this. Though students did expect Protection Poker to have benefits, they were divided in their expectations. Out of the 28 that responded to the open-ended question “How do you think playing PP will influence the product?” 10 stated that they did not expect much influence. Of those that did expect an influence, the majority (11) expected it to improve security awareness.

Other expectations included identifying the most important parts regarding security (4), a more secure product (3), discussions on security (2) and agreement on security issues in the group (2). Those that explained why they did not expect an impact from playing the game, explained that this was because of limited security issues in their project. Open-ended responses to the question “How do you believe software security is important to your project?” confirm our observations that only two of the six groups had clear security issues that needed to be dealt with, while four groups had limited attack surfaces or assets of little value, thus having limited security needs. Because it is likely that the limited security needs of projects influenced the usefulness of the technique when it comes to security, we additionally looked at the responses from the two groups that had security issues (ten responses) in isolation. We found that for the question on whether Protection Poker will improve the security of the product, all five students that agree that using Protection Poker improved security come from these two groups. When it comes to reduction of security defects, the students from the groups with security issues are more positive than the others, however, also these students in general do not agree that they experienced such a reduction from playing Protection Poker. Note that the graduate students in the software security are more positive regarding the effect of playing Protection Poker on security. These students both had more security competence and a project where security was an important part because patient medical records are involved.

In spite of limited need for security, the questionnaire responses indicate that students still ended up being quite positive regarding the usefulness of the game. Positive aspects of the game were discussed in the group interviews, and these can shed some light on what the students found useful. Overall, the students were positive to security and see the need for it in the general case. They explained that they learned many things from playing. This included knowledge about security (assets, attack surface, easy to overlook security issues). However, other more general insights were more often pointed out, such as gaining experience in group discussions, making decisions, coming to consensus, etc., and that they learned things about their own software projects and how it was understood by other group members. As stated by one student in the group interviews:

I think it is a good game, I think it works fine, but I don't think I got that much out of it as I could have, and I could have learned more about the different parts of Protection Poker and software security if I had a game or a project with more security issues.

## 6. Discussion

Through our interaction with the industry and in the student projects we find that people generally react positively to Protection Poker when we introduce the game, but there is room for improvements. As can be seen from [Tables VI and VII](#), industry representatives and students that used Protection Poker in different types of projects ended up experiencing many of the same benefits and challenges with Protection Poker. The main differences are as follows:

- Industry representatives and students doing a security course are more positive on the effects Protection Poker has on security.
- Industry representatives find Protection Poker more difficult than the students (to learn and use).
- Less challenges related to the discussion were observed with industry representatives.
- Industry representatives identified challenges that were not relevant for the student projects, e.g. challenges with scalability and with integrating the playing of Planning Poker with other planning activities in industry settings.

Many of the challenges identified with Protection Poker in this study overlap with challenges identified in literature when it comes to integrating security activities in agile development (see Section 2.2); these include challenges on:

- running the meetings (Cruzes *et al.*, 2018): e.g. challenging discussions and group dynamics (C4-5 in Table VII);
- documentation (Cruzes *et al.*, 2018): e.g. that the end result is not concrete enough (C16) and that key aspects from the discussion is lost (C11);
- integrating security techniques into the development work (Turpe and Poller, 2017): e.g. gaining impact from playing (C1) and integrate with other activities and way of work (C13, C14);
- priorities (Terpstra *et al.*, 2017): e.g. making the time needed acceptable (C3, C17); and
- awareness and knowledge (Terpstra *et al.*, 2017): e.g. making security terms understandable for the players (C9-10) and ensuring confidence in the results (C6).

Because of the overlap in types of challenges found, many of the issues that end up being challenging with Protection Poker may not be because of this particular technique, but rather point to more general challenges with this type of work. Thus, finding ways to tackle such challenges with Protection Poker can be useful input also to other techniques in this domain.

Based on the findings from this study, we would point at four major issues that need to be improved on Protection Poker:

- (1) making the time needed to play Protection Poker more acceptable for the teams;
- (2) ensuring impact from playing Protection Poker on the security of the end product;
- (3) better integration of Protection Poker with project-planning activities; and
- (4) clarifying the scenarios for better adoption of Protection Poker in a project.

The students in the capstone projects ended up finding that Protection Poker did not improve the security of the product (C1). Even though the professionals that participated in this study and the graduate students were more positive on this aspect of Protection Poker, such a feedback on the technique needs to be addressed, as a goal of improved security is the main reason for investing time in performing such a technique in the team. Additionally, we got the feedback both from students and professionals that playing Protection Poker took too much time (C3-4, C17), and professionals pointed out the difficulties in integrating such a time-consuming activity into existing planning activities in the projects (C13). As time and budget is considered important factors that influence adoption of software security practices (Kanniah and Mahrin, 2016) (see Section 2.2), it is important to address challenges related to the time needed. Results additionally point to Protection Poker being more useful in some of the teams than in others; thus, teams considering adopting Protection Poker should be aware of the factors that can impact the usefulness of the technique for their particular situation. In the following we discuss how to improve these parts of the game. An overview of the approach we are suggesting can be found in Figure 3.

#### 6.1 Reduce time needed to start using Protection Poker

Other studies have shown that the time needed to play Protection Poker is reduced after it has been used by the team for some time (Williams *et al.*, 2010). Still, it is important to ensure that teams considering to adopt Protection Poker are not put off by the time it takes to start

---

using Protection Poker the first time (C3). Based on our experiences in these studies, the long time it takes to begin using Protection Poker is because of the following issues:

- the time needed to learn the technique itself;
- the time needed to become familiar with the terms “asset”, “asset value” and “exposure”/“ease of attack”;
- the need to calibrate the scale; and
- the need to play about all asset values related to a feature.

---

**529**

There is nothing that points to Protection Poker being more time consuming to learn than other techniques, in fact one benefit identified is that Protection Poker is easy to learn (B2 in Table VI). In the study, some challenges on understanding the terms were identified (C9-10). However, this would probably be the case also for other security techniques if the players are unfamiliar with software security (as was the case for most of the students in the capstone projects). Though Protection Poker may benefit from terms that are even easier to use and explain, the terms “asset” and “exposure” (or variations thereof) are common in security techniques. Thus, it is likely that the main improvements related to Protection Poker and the time it starts to use the technique would be related to calibration and playing about asset value.

In the capstone projects, we skipped part of the calibration to save time. Although we observed that this worked well in most of the groups, we got the feedback from the students that they would recommend taking the time to do a full calibration in future projects. Identifying and prioritizing assets was something that the capstone project students in general found useful (B7), however, there were challenges associated with doing this as part of playing Protection Poker: it took time and there was the concern that if this was not done at a similar granularity for the whole project this might skew the prioritization one ended up making through the game (C8).

There are various options to reduce the time needed to do calibration and asset identification. Teams could:

- decouple asset identification from the playing of Protection Poker;
- let one or two people perform calibration as a preparation; and
- drop the division into assets and exposure/ease of attack altogether and play about only one issue per feature, e.g. “How easy is it to misuse this feature to get to/harm important assets?”

Decoupling of asset identification from the playing of Protection Poker was suggested by both students in capstone projects and by professional developers in one company. As this is likely to speed up playing, as well as limit the challenge related to granularity of assets (C8), we suggest that teams take this approach. Regarding calibration, we expect that this could be done as a preparation, as we did not have major challenges in most teams where full calibration was dropped. This would allow teams to spend less time on calibration but still have a scale that is calibrated for their specific project (B6). Regarding the alternative of dropping the division into assets and exposure altogether, we have no data to indicate whether or not this would be an improvement to the game. Thus, we do not recommend that teams take this approach. However, we would welcome more research into what are the most important questions to ask in a security analysis task. We observed that players mixed up the terms assets and exposure and had challenges in understanding these terms (C9-10). Thus, it may be just as easy for them to drop this division and consider them together. However, one important thing that the capstone project students liked was to get an

overview of the assets for the project (B7) and if going in this direction one would lose that benefit of playing Protection Poker.

### *6.2 Gaining impact from playing Protection Poker*

Apart from limited security needs in four of the capstone projects (C2), we do not know what factors that potentially made it difficult to use the results from the Protection Poker game in the development. In the capstone projects, it was up to the students to use the results from playing in any way they found fit. We did not follow up on how they used the results and provided no specific guidance on how to do this. One potential issue is the limitation pointed out by one student in the group interview that the game does not include anything on *how* such an attack can be mitigated. If students lack this knowledge it can be difficult to understand what can be done to reduce the risk associated with what they consider high risk functionality in the software. The results from playing Protection Poker is a prioritization, something the students found to be an important benefit (B7); however, turning this prioritization into actual development tasks is not necessarily straightforward, especially if the rationale for the scores is lost because of limited note taking (C11).

A major benefit of the Protection Poker technique is that it involves all team members in useful discussions (B4-5). However, for these discussions to have an impact on the end product, it is important that key aspects from these discussions are documented (C11) and made into actionable tasks (C16). To encourage the documentation of more aspects from the discussions, we recommend that teams have team collaboration software open during the playing of Protection Poker, and that one person (e.g. the facilitator, the security champion of the team, or someone else appointed this task) is responsible for adding important issues along the way. This includes open issues that need to be investigated further, suggestions for mitigations, attack vectors that should be considered, new assets identified, opinions on the whether the risk is acceptable or not and ideas for tests. This way one ensures that the team not only is left with a score after the discussions but also has the rationale behind the score. After the session where Protection Poker is played, it is essential that someone is responsible for making decisions based on the result from playing Protection Poker and enforcing these decisions by adding development tasks.

### *6.3 Integrate Protection Poker with project-planning activities*

A general impression, after observing the playing of Protection Poker and talking with practitioners, is that few teams would be willing to adopt Protection Poker for every iteration. Many of the main benefits from the technique as identified in this study, such as good discussions (B4), increased security awareness (B9) and overview of system (B1), is not dependent on playing Protection Poker for every feature. However, if not playing about every feature then teams need other ways to ensure that security is addressed in a holistic manner, and teams may lose the benefit of having regular security discussions in the full team. One option to solve this challenge is to play Protection Poker less regularly, and for more high-level features. Instead of playing Protection Poker for every feature that is to be implemented in an iteration, one could play Protection Poker for the main features in each epic (more overall groups of functionalities). Playing Protection Poker for each epic comes with the benefit that one can more easily use the result of playing Protection Poker as input to effort estimation and prioritization. Alternatively, teams would need some criteria as to when a round of Protection Poker should be played. Teams could also decide to use Protection Poker entirely as an awareness and training tool at various points throughout the project, discussing the features that are more relevant for security in that stage of the project. Then other techniques are necessary to ensure the elicitation of security requirements.



#### 6.4 Criteria for adopting Protection Poker in a project

The results from this study show that Protection Poker is not that useful for projects with very limited security issues, because of either limited attack surface or few assets of any particular value. This type of projects is probably not as prominent in development companies as in our case with capstone projects. Still, our results point to the need for some kind of criteria to evaluate whether there are enough security issues in a project to justify the effort needed to play Protection Poker.

The discussions were a main source of the benefits from playing Protection Poker and a source of challenges, especially concerning time. Because of team dynamics issues, the teams experienced the playing of Protection Poker quite differently. Protection Poker initially aims to support good discussions, and the voting involved when putting out a card is a way to ensure that all team members' opinions are made visible. However, the goal of reaching a consensus is not realistic in many settings. Teams need to be made aware that this is challenging, and not necessarily a strict goal. Though it is not beneficial to have everybody always agree, playing Protection Poker with participants that *never* agree, or always need to be right, is challenging. Based on the results from this study one can assume that how well Protection Poker will perform in the team, and especially the efficiency of the discussions, is highly influenced by the team dynamics. Knowledge of team dynamics could thus be one factor to consider when deciding if Protection Poker would be a good technique for a particular team.

One reason why Protection Poker takes time is that it is recommended that the full team participates in the playing. Having only one or two persons make an evaluation of assets related to a feature, their value and the exposure related to a feature would be more efficient in terms of time. However, it would be hard to get the same awareness raising in the whole team related to security (B9) (Williams *et al.*, 2010) with such an approach. Thus, teams that already have a high awareness and knowledge about software security may not need to spend the extra time on playing Protection Poker in the full team. However, if awareness and knowledge raising is needed on security, playing Protection Poker could be considered a type of security training for the whole team, and thus the extra time may be well worth the effort in the long run. Organizations and teams should consider this before deciding whether to invest time in playing Protection Poker.

### 7. Threats to validity

This study involves both students and professional software developers; however, most of the data come from student projects. By performing the major parts of the study in a university setting we were able to control the setup of the study in a way that would be difficult to do with companies. Additionally, we had the ability to collect more data, as the time students needed to invest in the data-collection activities was less of a concern compared to what would be the case for professional developers. This allowed us to use several data-collection methods to increase confidence in the results. Still, performing the study with students has its drawbacks as these have different experiences and are in a different context than professional software developers.

Research has shown that students in the later parts of their studies can be used with success in studies instead of professional software developers in some cases, namely, for understanding dependencies and relationships in software engineering (Höst *et al.*, 2000) and for requirements selection (Svahnberg *et al.*, 2008). The topic of this study is related to, but not identical to, those studies. We do not claim that the results from our study can be generalized to software developers in general but believe it to be likely that many of the same issues that we found would apply also in professional settings, in particular because many professionals in small

and medium-sized development organizations would also be considered novices when it comes to Protection Poker and have limited software security training (Jaatun *et al.*, 2015). However, the context would be different. Although the students in the capstone projects did have an external customer and the aim of the course is to have a setting that is as similar as possible to a real development project, the students had some concerns that professionals would not have (e.g. writing the report and getting a good grade) and this may have impacted the results. Their development projects were also likely to be simpler and with fewer security concerns than what many professional developers would likely encounter. From the results of our study we can see that many of the same benefits and challenges were observed with both students and professionals, but it is important to note that there were main differences as well. Thus, a study with more participation from professionals may have yielded different results on some aspects.

In the capstone project study, it is difficult to separate the effect of the technique itself from other factors, such as motivation, skills, group dynamics and our influence as researchers. In particular, having researchers act as facilitators constitutes a threat to validity that may influence the process and the results and make the study harder to replicate. We have aimed to be aware of the impact of the context throughout the capstone project study. One way we did this is by having the first author be supervisor of one student group. Additionally, we made sure we reflected on our role as researchers and took this into account in the analysis (reflection on our influence as researchers was part of the template for observation notes). As part of this, we made it clear for students that their opinion on Protection Poker would not have any impact on their grade in the course. We as researchers did not have any influence on the grades the students got, except for giving some input to evaluators for the group where the first author acted as supervisor.

In the events with professionals, our ability to collect rich data was reduced, and these events were more limited in time than what was the case with students. The companies were recruited based on existing contacts. Additionally, the response rates on the questionnaires were sometimes rather low. Both these factors may lead to more positive responses regarding Protection Poker than what we may have gotten in other companies and with more responses, as it is likely that the more positive companies and participants when it comes to security are willing to participate. The participants in the session in the conference mainly had roles and background relating to security, and limited development background. Security people have a potentially important role in promoting and supporting security work in software projects (Tøndel *et al.*, 2017), thus the opinions of representatives with this role is relevant to consider. However, their opinions and perspectives may differ from developers. We did not aim to analyse differences between responses from different groups of professionals, as the number of questionnaire responses was not large enough to justify such analysis.

In the graduate level course, students performed the Protection Poker activity on their own and reported the results. They were not aware they would be completing a questionnaire when they completed the assignment, and the student were made aware questionnaire was anonymous. When asked about their participation on the Protection Poker part of the assignment, approximately 10 per cent of the students indicate they did not participate fully yet they answered the questionnaire. There were three other parts of the assignment, and sometimes the students divided up the work throughout the group.

## 8. Conclusion

Protection Poker is a collaborative technique for risk estimation that is particularly suited for agile development teams. As of now we are not aware that Protection Poker ends up being adopted by teams. This study has identified both benefits and challenges with Protection Poker. It suggests how to tackle the main obstacles to adoption of the technique,

including ways to address the challenge that teams may find that Playing Protection poker takes more time than they are willing to spend, at least in every iteration. Additionally, it points to the importance of finding ways to ensure playing Protection Poker ends up having an impact in form of improved security of the end product. To tackle this, this paper suggests increasing preparation activities and ensuring more documentation and follow-up from the discussions that take place during the playing.

#### Notes

1. Available at: <https://sites.google.com/a/ncsu.edu/csc515-software-security/>
2. Available at: <https://openmrs.org/>

#### References

- Ajzen, I. and Fishbein, M. (1980), *Understanding Attitudes and Predicting Social Behaviour*, Pearson.
- Alsaqaf, W., Daneva, M. and Wieringa, R. (2017), "Quality requirements in large-scale distributed agile projects – a systematic literature review", *International Working Conference on Requirements Engineering: Foundation for Software Quality*, Springer.
- Baca, D., Boldt, M., Carlsson, B. and Jacobsson, A. (2015a), "A novel security-enhanced agile software development process applied in an industrial setting", *2015 10th International Conference on Availability, Reliability and Security (ARES)*, IEEE.
- Baca, D., Boldt, M., Carlsson, B. and Jacobsson, A. (2015b), "A novel Security-Enhanced agile software development process applied in an industrial setting", *2015 10th International Conference on Availability, Reliability and Security*, 24-27 August, pp. 11-19.
- Beck, K., Beedle, M.V., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A. and Jeffries, R. (2001), "Manifesto for agile software development".
- Caralli, R.A., Stevens, J.F., Young, L.R. and Wilson, W.R. (2007), "OCTAVE allegro: improving the information security risk assessment process (CMU/SEI-2007-TR-012 ESC-TR-2007-012)", Software Engineering Institute at Carnegie Mellon University.
- Caroli, P. and Caetano, T. (2015), "Fun Retrospectives – Activities and ideas for making agile retrospectives more engaging", Leanpub, available at: [www.caroli.org/en/book-fun-retrospectives](http://www.caroli.org/en/book-fun-retrospectives)
- Chandra, P. (2008), "Software assurance maturity model", A guide to building security into software development v1.0, OWASP Project.
- Cruzes, D.S., Jaatun, M.G., Bernsmed, K. and Tøndel, I.A. (2018), "Challenges and experiences with applying Microsoft threat modeling in agile development projects", *Australasian Software Engineering Conference (ASWEC)*, Adelaide.
- Cybenko, G. (2006), "Why Johnny can't evaluate security risk", *IEEE Security and Privacy*, p. 5.
- Davis, F.D. (1985), *A Technology Acceptance Model for Empirically Testing New End-User Information Systems: Theory and Results*, MA Institute of Technology.
- Davis, F.D. (1989), "Perceived usefulness, perceived ease of use, and user acceptance of information technology", *MIS Quarterly*, Vol. 13 No. 3, pp. 319-340, doi: [10.2307/249008](https://doi.org/10.2307/249008).
- Deleersnyder, S., Win, B.D. and Glas, B. (2017), "Software assurance maturity model – how to guide – a guide to building security into software development", 1.5 ed., OWASP.
- Dingsøyr, T., Moe, N.B., Fægri, T.E. and Seim, E.A. (2018), "Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation", *Empirical Software Engineering*, Vol. 23 No. 1, pp. 490-520.
- Fenz, S. and Ekelhart, A. (2010), "Verification, validation, and evaluation in information security risk management", *IEEE Security and Privacy*, pp. 58-65.

- Geer, D. (2010), "Are companies actually using secure development life cycles?", *Computer*, Vol. 43 No. 6, pp. 12-16.
- Gerber, M. and Von Solms, R. (2005), "Management of risk in the information age", *Computers and Security*, Vol. 24, pp. 16-30.
- Grenning, J. (2002), "Planning poker or how to avoid analysis paralysis while release planning", *Hawthorn Woods: Renaissance Software Consulting*, Vol. 3, pp. 22-23.
- Höst, M., Regnell, B. and Wohlin, C. (2000), "Using students as subjects – a comparative study of students and professionals in lead-time impact assessment", *Empirical Software Engineering*, Vol. 5 No. 3, pp. 201-214, available at: <https://link.springer.com/article/10.1023/A:1026586415054>
- Howard, M. and Lipner, S. (2006), *The Security Development Lifecycle: Process for Developing Demonstrably More Secure Software*, Microsoft Press.
- ISO/IEC (2011), "ISO/IEC 27005: 2011 Information technology–security techniques–information security risk management".
- Jaatun, M.G. and Tøndel, I.A. (2008), "Covering your assets in software engineering", *The Third International Conference on Availability, Reliability and Security, IEEE, Barcelona*.
- Jaatun, M.G. and Tøndel, I.A. (2016), "Playing protection poker for practical software security", *International Conference on Product-Focused Software Process Improvement, Springer*.
- Jaatun, M.G., Cruzes, D.S., Bernsmed, K., Tøndel, I.A. and Røstad, L. (2015), "Software security maturity in public organisations", in *International Conference on Information Security*, Springer, Cham, pp. 120-138, available at: [https://link.springer.com/chapter/10.1007/978-3-319-23318-5\\_7](https://link.springer.com/chapter/10.1007/978-3-319-23318-5_7)
- Jourdan, Z., Rainer, R.K., Marshall, T.E. and Ford, F.N. (2010), "An investigation of organizational information security risk analysis", *Journal of Service Science*, Vol. 3, p. 10.
- Kanniah, S.L. and Mahrin, M.N. (2016), "A review on factors influencing implementation of secure software development practices", *International Journal of Computer and Systems Engineering*, p. 10, available at: <http://doi.org/10.5281/zenodo.1127256>
- Khaim, R., Naz, S., Abbas, F., Iqbal, N. and Hamayun, M. (2016), "A review of security integration technique in agile software development", *International Journal of Software Engineering Applications*, Vol. 7 No. 3, pp. 49-68.
- Li, L. (2010), "A critical review of technology acceptance literature", *Southwest Decision Sciences Institute Conference*.
- Mcgraw, G. (2004), "Software security", *IEEE Security and Privacy Magazine*, Vol. 2, pp. 80-83.
- Mcgraw, G. (2006), *Software Security: Building Security In*, Addison-Wesley Professional.
- Mcgraw, G., Miguez, S. and West, J. (2016), "Building security in maturity model (BSIMM7)", Cigital.
- MICROSOFT (2012), "Security development lifecycle for agile development", available at: <https://msdn.microsoft.com/en-us/library/windows/desktop/ee790621.aspx>
- NIST (2010), "Guide for applying the risk management framework to federal information systems – a security life cycle approach", R1 ed.
- Nyford, J. and Kajko-Mattsson, M. (2008), "Integrating risk management with software development: state of practice", *International MultiConference of Engineers and Computer Scientists, IAENG, Hong Kong*.
- Odzaly, E.E., Greer, D. and Stewart, D. (2018), "Agile risk management using software agents", *Journal of Ambient Intelligence and Humanized Computing*, Vol. 9 No. 3, pp. 823-841, available at: <https://doi.org/10.1007/s12652-017-0488-2>
- Oueslati, H., Rahman, M.M. and Ben Othmane, L. (2015), "Literature review of the challenges of developing secure software using the agile approach", *2015 10th International Conference on Availability, Reliability and Security (ARES), IEEE*.
- Rhee, H.-S., Ryu, Y.U. and Kim, C.-T. (2012), "Unrealistic optimism on information security management", *Computers and Security*, Vol. 31, pp. 221-232.

- Sulaman, S.M., Weyns, K. and Höst, M. (2013), "A review of research on risk analysis methods for IT systems", *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, ACM, pp. 86-96.
- Svahnberg, M., Aurum, A. and Wohlin, C. (2008), "Using students as subjects-an empirical evaluation", in *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ACM, pp. 288-290, available at: <https://dl.acm.org/citation.cfm?id=1414055>
- Tavares, B.G., Da Silva, C.E.S. and de Souza, A.D. (2017), "Risk management analysis in scrum software projects", *International Transactions in Operational Research*, Vol. 26 No. 5.
- Terpstra, E., Daneva, M. and Wang, C. (2017), "Agile practitioners' understanding of security requirements: insights from a grounded theory analysis", *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, IEEE.
- Tøndel, I.A. (2018), "Results from questionnaires on protection poker".
- Tøndel, I.A., Jaatun, M.G., Cruzes, D.S. and Moe, N.B. (2017), "Risk centric activities in secure software development in public organisations", *International Journal of Secure Software Engineering*, Vol. 8 No. 4, pp. 1-30.
- Tøndel, I.A., Jaatun, M.G., Cruzes, D.S. and Oyetoan, T.D. (2018), "Understanding challenges to adoption of the protection poker software security game", *2nd International Workshop on SEcURITY and Privacy Requirements Engineering (SECPRE 2018)*, Barcelona.
- Tøndel, I.A., Line, M.B. and Johansen, G. (2015), "Assessing information security risks of AMI: What makes it so difficult?", *1st International conference on Information Systems Security and Privacy 2015, Angers*.
- Türpe, S. and Poller, A. (2017), "Managing security work in scrum: tensions and challenges", *International Workshop on Secure Software Engineering in DevOps and Agile Development, CEUR-WS, Oslo*.
- Venkatesh, V. and Davis, F.D. (1996), "A model of the antecedents of perceived ease of use: Development and test", *Decision Sciences*, Vol. 27 No. 3, pp. 451-481.
- Williams, L., Gegick, M. and Meneely, A. (2009), "Protection poker: Structuring software security risk assessment and knowledge transfer", *International Symposium on Engineering Secure Software and Systems*, Springer.
- Williams, L., McGraw, G. and Miguez, S. (2018), "Engineering security vulnerability prevention, detection and response", *IEEE Software*, Vol. 35 No. 5, pp. 76-80.
- Williams, L., Meneely, A. and Shipley, G. (2010), "Protection poker: the new software security game", *IEEE Security and Privacy Magazine*, Vol. 8 No. 3, pp. 14-20.

**Corresponding author**

Inger Anne Tøndel can be contacted at: [inger.anne.tondel@ntnu.no](mailto:inger.anne.tondel@ntnu.no)

For instructions on how to order reprints of this article, please visit our website:

[www.emeraldgroupublishing.com/licensing/reprints.htm](http://www.emeraldgroupublishing.com/licensing/reprints.htm)

Or contact us for further details: [permissions@emeraldinsight.com](mailto:permissions@emeraldinsight.com)


## **Paper D: ‘Towards a Conceptual Framework for Security Requirements Work in Agile Software Development’**

A written permission to include this material in its published form [30] has been obtained from IGI Global.


D

# Towards a Conceptual Framework for Security Requirements Work in Agile Software Development

Inger Anne Tøndel, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

 <https://orcid.org/0000-0001-7599-0342>

Martin Gilje Jaatun, SINTEF Digital, Oslo, Norway

 <https://orcid.org/0000-0001-7127-6694>

D

## ABSTRACT

Security requirement work plays a key role in achieving cost-effective and adequate security in a software development project. Knowledge about software companies' experiences of security requirement work is important in order to bridge the observed gap between software security practices and security risks in many projects today. Particularly, such knowledge can help researchers improve on available practices and recommendations. This article uses the results of published empirical studies on security requirement work to create a conceptual framework that shows key concepts related to work context, this work itself and the effects of this work. The resulting framework points to the following research challenges: 1) Identifying and understanding factors important for the effect of security requirements work; 2) Understanding what is the importance of the chosen requirements approach itself, and; 3) Properly taking into account contextual factors, especially factors related to individuals and interactions, in planning and analysis of empirical studies on security requirements work.

## KEYWORDS

Agile Software Development, Conceptual Framework, Empirical Studies, Literature Review, Security Requirements, Software Security

## INTRODUCTION

In today's interconnected world, we would claim that software security is an aspect to consider in most software development projects. Currently, agile development methodologies are prominent in software development. Such methods are used even for large scale development (Dikert, Paasivaara, & Lassenius, 2016) and for security critical and safety critical software (Hanssen, Stålhane, & Myklebust, 2018; Heeager & Nielsen, 2018; Oueslati, Rahman, & ben Othmane, 2015). Thus, good ways of working with security within an agile development paradigm is necessary.

There has been done a lot of work on suggesting ways to deal with software security in agile development projects, including proposals for integrating security into agile methodologies like XP

DOI: 10.4018/IJSSSP.2020010103

Copyright © 2020, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.



(Aydal, Paige, Chivers, & Brooke, 2006) and Scrum (Pohl & Hof, 2015). In 2009 the Microsoft Security Development Lifecycle (SDL) (Howard & Lipner, 2006; Microsoft, n.d.) was released in a version specific for agile development (Agile SDL) (Microsoft, 2009), and abuser stories have for some time been a suggested way of representing security requirements within agile development (Peeters, 2005). Additionally, there exist method-agnostic approaches to software security that should be possible to integrate with agile development, such as the touchpoints for software security (McGraw, 2004, 2006), the Building Security in Maturity Model (BSIMM) (McGraw, Miguez, & West, 2018) and the OWASP Software Assurance Maturity Model (SAMM) (OWASP, n.d.). There thus seems to be no lack of methods for doing software security work also within an agile paradigm. Still, many have observed that security is frequently not given proper attention in software development projects today (Tøndel, Jaatun, Cruzes, & Moe, 2017; Nicolaysen, Sassoon, Line, & Jaatun, 2010; Terpstra, Daneva, & Wang, 2017).

As is well communicated by the abovementioned software security approaches, software security should be an integrated part of development and have a role in the various software development activities, including requirements, design, coding, testing, deployment and operations. Security is not something that can be successfully added on as an afterthought, but should be built into the system from the beginning. This however means that the total number of suggested security activities can be quite overwhelming. It is possible for projects to spend a lot of effort on security, even overspending, if not properly addressing the security needs. Thus, we consider security requirements work as foundational to achieving cost-effective security in a project.

In this article, we define software security requirements work as activities performed in relation to a software development project to: 1) decide whether and how to identify security needs, risks or requirements for a project; 2) do the requirements elicitation; 3) communicate the identified security needs, risks or requirements, and; 4) integrate these and make priorities related to them in development. By agile development we mean software development that in large part is guided by the Agile principles, as outlined in the Agile Manifesto (Beck et al., 2001), including various methods such as Scrum and XP. Compared to a waterfall development approach, requirements management in agile development is “far more temporal, interactive and just in time” (Leffingwell, 2010). Additionally, the need for requirements prioritization can be considered to be built into the approach; “[w]e admit up front that we can’t implement (nor even discover) all potential requirements” (Leffingwell, 2010). Security is only one of the types of requirements a development project needs to consider. When negotiating the three variables cost, schedule and requirements (Leffingwell, 2010), requirements may be modified or dropped altogether.

There exist few empirical studies on how security requirements are handled in software development projects (Terpstra et al., 2017), thus “[h]ow practitioners in the field think about security requirements and how they devise their processes of coping with the issues these requirements pose, is hardly known” (Terpstra et al., 2017). Empirical studies of software security practices within agile development can help us understand what makes companies and projects adopt (or not adopt) software security practices, how different practices may help, what works well and what is challenging in certain contexts, etc. Such knowledge is important in order to bridge the observed gap between software security practices and software security needs and risks in many projects today. In particular, such knowledge can help researchers improve on existing support provided to agile development projects in terms of available practices and recommendations.

In this article, we aim to improve understanding of security requirements work within an agile development approach. Our study aims to answer the following research questions:

- What factors are found in current empirical research to influence and/or characterize security requirements work in agile projects in an industry setting?
- How do these factors impact the security requirements work and its effect?

We build on published empirical studies that cover security requirements work, emphasizing results from studies performed in an industry setting. The findings from these studies are used to build a conceptual framework that shows important concepts that impact and/or characterize security requirements work, and the relations between these concepts. The conceptual framework is a step towards a comprehensive understanding of security requirements work in agile projects, and can act as a basis for further research on this topic, both in prioritizing which research studies to undertake, and as input to planning and analysis in future empirical studies within this topic.

The remainder of this article is structured as follows. It starts by giving an overview of literature reviews related to security requirements work in agile development and introducing two previously suggested conceptual framework related to this type of work. Then it moves on to explaining the method used to identify empirical sources and construct the conceptual framework. Following the method description, the article introduces the selected studies and the concepts derived from these studies before it presents the resulting conceptual framework. Then it discusses the validity of the conceptual framework and its implications for research. The article ends with a summary of the main conclusions of the article and introduces future work.

## RELATED WORK

Since the publication of the Agile Manifesto in 2001 (Beck et al., 2001), many researchers and practitioners have worked on how to include security into agile software development. These discussions started quite early; Systematic Literature Reviews (SLRs) identify papers from the very beginning discussing security in relation to agile development (Saldanha & Zorzo, 2019). One example of such a paper is Beznosov's suggestions from 2003 on how to integrate security and eXtreme Programming (XP) (Beznosov, 2003). Although security in agile development has been a topic of discussion and research since then, and a substantial amount of literature is available on the subject (Bishop & Rowland, 2019), many authors point to the need for more research to better understand and solve the challenges today's software development projects are facing when it comes to security (Bishop & Rowland, 2019; Saldanha & Zorzo, 2019; Villamizar, Kalinowski, Viana & Fernández, 2018).

Lately, several literature reviews have been performed regarding agile software development and security, and even specifically on security requirements in agile development. In this section, we give an overview of these literature reviews and position this article in relation to these reviews. Additionally, this section describes conceptual frameworks that have already been developed related to security requirements in agile development.

### Overview of Systematic Literature Reviews

In 2015, Oueslati et al. performed a literature review aimed at identifying "challenges of developing secure software using the agile approach" (Oueslati et al., 2015). Fourteen challenges were identified, and the challenges were categorized into five categories: "Software development life-cycle challenges" concerned security activities not being included in agile methods, and difficulty integrating security in every iteration due to short iteration times. "Incremental development challenges" were related to dealing with frequent changes. "Security assurance challenges" were related to documenting and testing the system in a manner expected for security assurance. "Awareness and collaboration challenges" included neglect of security requirements, lack of experience and security awareness, and challenges separating the developer and reviewer roles. "Security management challenges" concerned how added costs and a lack of incentives resulted in security not being prioritized.

In 2016, Khaim et al. studied what approaches are suggested for software security in agile, the role of the security expert in these approaches, and what kind of challenges emerge when integrating security and agile development (Khaim et al., 2016). They found that half of the studies consider integration of security into agile in a general way, 15% consider integration into Scrum, 23% XP and

12% FDD. In over half (54%) of the papers they studied, the security expert role was not specified, and the impression of Khaim et al. is that those papers that include the security expert role do not do this in a clear way and in a way that ensures involvement throughout development. Khaim et al. identified a long list of challenges, including separating the software developer and security expert roles, security assurance in the case of continuously changing code, documentation needs, refactoring, lack of security experience of developers, tracking security requirements in case of frequent changes, neglecting security requirements, unaware customers, etc.

In 2017, Alsaquaf et al. performed an SLR on quality requirements in large scale agile development in order to identify practices and proposed approaches to cope with quality requirements challenges in large scale distributed agile development (Alsaquaf, Daneva & Wieringa, 2007). Security was considered a type of quality requirement. They found that, despite many available approaches, none of the approaches they identified had been “tried out in real life settings” (Alsaquaf et al., 2017). Challenges were identified related to the techniques available (no widely accepted techniques; inadequacy of the existing techniques; traceability), the priorities made (functionality is prioritized; ignore some types of requirements; validated late; insufficient analysis) and related to the Product Owner (lack of knowledge; workload; availability; dependence). They pointed out that the challenges in large part relate to agile-specific practices and that “some characteristics of agile [requirements engineering] pitched as strengths in agile textbooks (e.g. the role of the product owner, the use of user stories) can be considered in fact as inhibitors to engineering of [quality requirements]” (Terpstra et al., 2017).

In 2019, Bishop and Rowland analyzed literature in order to understand “the effect of security practices on software development agility” (Bishop & Rowland, 2019). Additionally, they provided a taxonomy that can be used to organize and summarize work on software security in agile. They divided papers into two main categories: phase focused and phase independent. The requirements phase was identified as the phase that had received the most research attention. Still, they pointed to a need for more research, and especially empirical research, to extend the current body of knowledge related to software security in agile development.

Additionally, 2018 and 2019 saw the publication of three literature studies that specifically considered security requirements in agile development. Both Saldanha & Zorzo (2019) and Villamizar et al. (2018) performed systematic mapping studies to understand the approaches taken to handle security requirements in agile development projects, and to assess the coverage of current research on this topic. Villamizar et al. found that most approaches are related to Scrum, and that most approaches address specification and elicitation of security requirements. Both studies found that solutions typically involve modifying the agile method or introducing new artefacts or guidelines to handle security. Saldanha & Zorzo however also point to the importance of security training and its possibility to impact the security level of the software. Both systematic mapping studies identify a lack of empirical research, including empirical evaluations of existing approaches to security requirements engineering in agile development. Other research gaps identified include tool support and verification and validation of security requirements (Villamizar et al., 2018). Muneer et al. performed a systematic literature review to compare “modern requirements management techniques with classic techniques for managing Non-Functional requirements (NFRs) in agile Software Methods”, focusing primarily on security requirements as a type of NFR (Muneer, Nadeem & Kasi, 2019). Their review concludes that modern techniques have the potential to overcome some of the method weaknesses identified.

### Objective of This Work

The work presented in this article is not an SLR, but a lighter and less comprehensive form of literature review with a goal to complement existing SLRs in this area. Previous SLRs have covered challenges related to security and agile (Oueslati et al., 2015), have identified the approaches covered in research literature (Khaim et al., 2016; Saldanha & Zorzo, 2019; Villamizar et al., 2018; Bishop & Rowland, 2019; Alsaquaf et al., 2017), the weaknesses and strengths of available approaches (Muneer et al., 2019), the effect on agility (Bishop & Rowland, 2019), and the role of the security

expert (Khaim et al., 2016). Most SLRs point to the need for more empirical research (Alsaquaf et al., 2017; Bishop & Rowland, 2019; Saldanha & Zorzo, 2019; Villamizar et al., 2018). We are however not aware of any SLR that gives an overview of what we can learn from the empirical research that has already taken place on security requirements work in agile development projects. This article is an attempt to fill this gap.

As a mean to give an overview of what we can learn from existing empirical research on this topic, this article combines previous findings into a conceptual framework. A conceptual framework can be defined as “a network, or ‘a plane’, of interlinked concepts that together provide a comprehensive understanding of a phenomenon or phenomena” (Jabareen, 2009). It is a product of theorization (Jabareen, 2009) and has particular benefits for designing studies as it “forces you to be selective to decide which variables are most important, which relationships are likely to be most meaningful, and, as a consequence, what information should be collected and analyzed at least at the outset” (Miles & Huberman, 1994). Thus, a conceptual framework based on existing empirical research can be used to guide future empirical research on security requirements in agile development, a type of study that is most needed. Conceptual frameworks can be “rudimentary or elaborate, theory-driven or commonsensical, descriptive or causal” (Miles & Huberman, 1994). However, in this article we take the advice from Jabareen (2009) to take an interpretative approach, rather than a causal/analytical approach; stating “[c]onceptual frameworks aim to help us understand phenomena rather than to predict them” (Jabareen, 2009).

### Existing Conceptual Frameworks Related to Security Requirements in Agile Development

We are aware of two existing attempts to create conceptual frameworks related to security requirements in agile development. These have a different foundation than the conceptual framework presented in this paper.

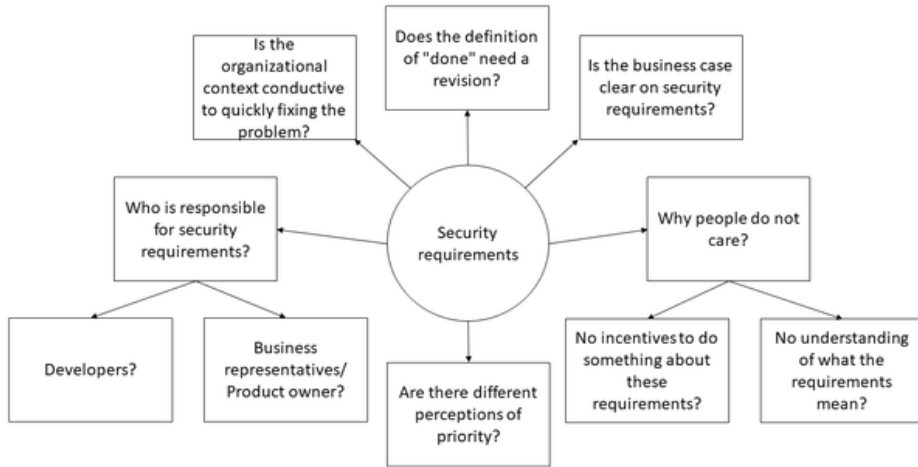
#### *Conceptual Framework on Security Requirements as Viewed by Agile Practitioners*

Terpstra et al. (2017) conducted a survey of practitioners’ postings on social media (LinkedIn) to discover how agile practitioners reason about security requirements and how they cope with this type of requirements. The analysis resulted in the identification of 21 concepts that indicate problems regarding security requirements in agile, and 15 coping strategies. The problems identified are varied, with central themes being the business value of security, the priorities that have to be made, the tendency that security gets lost in the process, and the lack of awareness and knowledge. The analysis additionally resulted in the development of a descriptive conceptual framework from the viewpoint of the development team. This conceptual framework has been redrawn in Figure 1. The boxes in this figure represent conceptual categories defined by Terpstra et al. that map to the figure in the following way:

- **Ownership of security requirements:** Who is responsible for security requirements? Developers? Business representatives/Product owner?
- **Definition of “done” (DoD):** Does the DoD need a revision?
- **Business case:** Is the business case clear on security requirements?
- **Attitude towards security requirements:** Why people do not care? No incentives to do something about these requirements? No understanding of what the requirements mean?
- **Organizational setup:** Is the organizational context conducive to quickly finding the problem?
- **Perceptions of priority:** Are there different perceptions of priority?

The conceptual framework developed by Terpstra et al. describe important influences and challenges with security requirements work in agile. It is however only based on one study.

Figure 1. Conceptual framework developed by Terpstra et al. (2017)



*Conceptual Framework Showing Important Categories When Integrating Security Requirements Into Agile Development*

Daneva & Wang (2018) performed a document analysis of seven “well documented agile secure development frameworks put forward by companies or non-profit industry organizations supported by companies.” Based on the practices that were part of these documents, they created the conceptual framework depicted in Figure 2. The central overarching category of the framework is “Absorb security requirements”, representing that “the development team absorbs the needs and the responsibility for

Figure 2. Conceptual framework developed by Daneva and Wang (2018)



engineering security requirements” (Daneva & Wang, 2018). What this means is represented by the other concepts in the framework:

- **Activities:** Introducing security activities (organizational and/or technical);
- **Artefacts:** Examples include abuser stories and risk assessments;
- **Roles:** Organizational or technical roles that could mirror agile specific roles (such as Scrum master or product owner), or adding security expertise to the team;
- **Competencies:** Having necessary competence, e.g. on security testing and secure architecture.

This conceptual framework presents important categories for security requirements work, as documented by key players. It however only describes current practice to the extent that the documented frameworks are used as written in these documents.

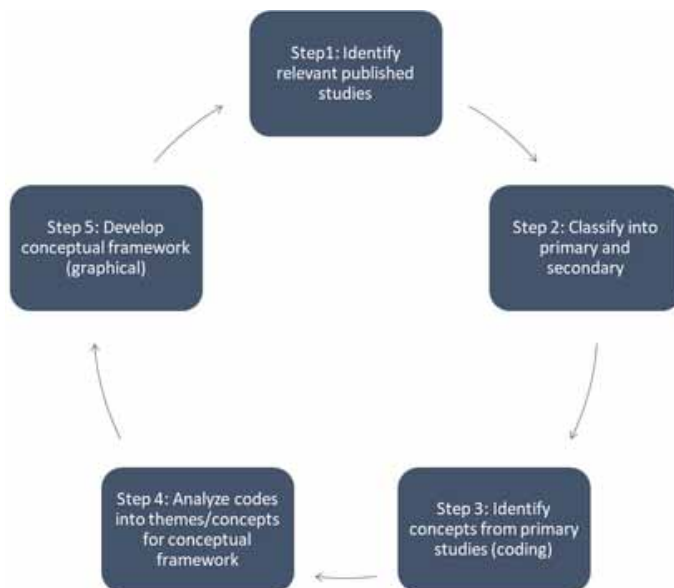
## RESEARCH METHOD

In order to construct a conceptual framework based on existing empirical studies, we performed a series of phases: identifying published studies to use as a basis for the framework, analyzing the published results of these studies to identify central concepts related to security requirements work, and synthesize these concepts into a conceptual framework. In the following we describe each of these phases in more detail. An overview of our approach is given Figure 3. As can be seen from this figure, our approach was cyclical. In all we did three iterations of this cycle.

### Identify Studies (Step 1)

Our goal in this step was to identify empirical studies covering aspects of security requirements work in agile development. We were primarily interested in studies performed in industry settings. We did

Figure 3. Method for constructing the conceptual framework



not aim for an SLR. Instead we used the following strategy to identify relevant studies: 1) identify relevant and recent SLRs and use them to find relevant papers, and; 2) supplement references from SLRs with searches on Google Scholar for more recent studies. We did this in three rounds and over a period of three years. The first round used the SLRs of Khaim et al. (2016) and Oueslati et al. (2015), in addition to searches on Google Scholar to identify more recent studies that were not covered by the published SLRs. We deliberately searched for empirical studies covering software security in agile, and not specifically security requirements work, because we use this term in a broader sense than eliciting and documenting security requirements. The second round used the SLR of Alsaquaf et al. (2017) as well as additional searches on Google Scholar. The third round used the SLRs of Bishop and Rowland (2019), Saldanha and Zorzo (2019), Villamizar et al. (2019) and Muneer et al. (2019).

We selected this lightweight approach to identifying empirical studies for two reasons: 1) we did not believe it necessary to repeat the identification of relevant publications already performed by other researchers, and 2) we did not have the goal to collect all relevant empirical evidence, but rather sufficient evidence to create a first version of a conceptual framework that could later be extended as new evidence is published.

The searching for and screening of literature was done by one researcher, and was done based on title and abstract. For the studies that seemed to fit the inclusion criteria (empirical studies of security requirements work in agile development), we further inspected the methods and results sections to determine if they were relevant for this study. In the end, we ended up with the papers listed in Table 1.

### **Analyze Study Results (Step 2 and Step 3)**

Our approach was inspired by Jabareen (2009), who proposed a method called conceptual framework analysis particularly suited for multidisciplinary phenomena. Jabareen's method is based on the grounded theory model, and consists of seven phases: 1) "Mapping the selected data sources", 2) "Extensive reading and categorizing of the selected data", 3) "Identifying and naming concepts", 4) "Deconstructing and categorizing the concepts", 5) "Integrating concepts", 6) "Synthesis, resynthesis, and making it all make sense" and 7) "Validating the conceptual framework" (Jabareen, 2009). We did not follow this method in detail, as we are not aiming for a multidisciplinary framework with as extensive sources as is recommended by Jabareen. Instead we used aspects of this method adjusted to our needs. In particular, we took the advice to read and reread sources, and to categorize them based on importance (Jabareen, 2009) and relevance (Maxwell, 2013). All the included studies (Table 1) were divided into two categories: primary studies and secondary studies. The first category consists of studies where security requirements work is a main part of the topic studied, the study is done in an industry setting and with adequate research method (including adequate method description). Secondary studies are studies with results that could be relevant for security requirements work, but where security requirements is only a minor part of the overall study or where the study is not adequately described, has obvious methodological limitations or is performed in a student setting. Only the primary studies were used to identify concepts, while results from the secondary studies were used where relevant to better understand and to validate/contradict the findings in the primary studies. Concepts were identified by reading the results and discussion part of the primary studies in detail, identifying any results related to security requirements work, and adding codes to these results.

### **Synthesize (Step 4 and Step 5)**

The method used for constructing a conceptual framework builds on the practical advice of Miles and Huberman (1994). To cite Miles and Huberman, "[s]etting out bins, naming them, and getting clearer about their interrelationships lead you to a conceptual framework." (Miles & Huberman, 1994). According to Miles and Huberman, conceptual frameworks are best done graphically, one should expect to do several iterations, and they suggest using prior theorizing and empirical research as important inputs. Additionally, they recommend that one should avoid the non-risk framework with only global level variables and "two-directional arrows everywhere" (Miles & Huberman, 1994).

Table 1. Overview of included papers and which SLRs that also include them

Identified Paper	Oueslati et al. (2015)	Khaim et al. (2016)	Alsaquaf et al. (2017)	Bishop & Rowland (2019)	Saldanha & Zorzo (2019)	Villamizar et al. (2019)	Muneer et al. (2019)
Adelyar & Norta (2016)				x			
Aydal et al. (2016)		x					
Baca & Carlsson (2011)				x	x		
Baca et al. (2015)		x			x		
Bartsch (, 2011)	x			x	x		x
Bellomo & Woody (2012)	x						
Fitzgerald et al. (2013)					x		
Ghani et al. (2014)				x	x	x	
Kongli (2006)	x	x		x		x	
Nicolaysen et al. (2010)							
Pohl & Hof (2015)		x					
Poller et al. (2017)				x			
Rajba (2018)				x			
Renatus et al. (2015)							
Rindell et al. (2016)					x		x
Sachdeva & Chung (2017)				x		x	x
Savola et al. (2012)		x					
Terpstra et al. (2017)				x	x		x
Tøndel et al. (2017)							
van der Heijden et al., 2018				x			
Williams et al. (2009)							
Williams et al. (2010)					x		

As will be seen from the following section, we made tables of concepts from the primary sources, as suggested by Jabareen. In making these tables we integrated and synthesized concepts, as suggested by Jabareen, in an iterative manner. Using Mind Manager, initial concepts identified from the primary studies were grouped into overall concepts. This was done first for the different study types identified, and then we combined concepts from the different study types into a modified and combined table of concepts that we used to create the conceptual framework. In creating the graphical conceptual framework, we looked again to the primary studies to identify what relations between the concepts were visible in those studies.

## CONCEPTS FROM EMPIRICAL STUDIES

In the following, we give an overview of the identified primary studies and the concepts identified from these studies. An overview of the primary studies can be found in Table 2, while an overview of secondary studies can be found in Table 3. We start with presenting studies that



**Table 2. Overview of primary studies**

Reference	Study Goal	Study Method
Adelyar & Norta (2016)	Identify challenges in the customer and developer practices	Interviews, 4 teams, team manager and 2-3 developers from each team
Baca et al. (2015)	Evaluate SEAP (more security resources in the team; incremental risk analysis)	Action research, one project, four versions, 8 development teams
Bartsch (2011)	Understand challenges and mitigations in security-critical agile development	Interviews (10 interviewees from 9 companies)
Nicolaysen et al. (2010) (interview part)	Understand whether software security was a specific concern in agile development	Six interviews with software developers
Poller et al. (2017)	Impact of external audit and training	Questionnaires, observations, document studies, interviews (15); 13 months
Terpstra et al. (2017)	Discover how agile practitioners reason about security requirements and how they cope with this type of requirements	Postings on LinkedIn, two discussion threads
Tøndel et al. (2017)	Identify risk-centric practices in software security	Interviews; 23 organizations
van der Heijden et al. (2018)	Identify security challenges in large-scale agile development	Interviews (ten interviews from five teams) in a financial organization
Williams et al. (2010)	Effect of using Protection Poker (a technique for security risk estimation)	Case study (observations, survey) in one team, 3 months

**Table 3. Overview of secondary studies**

Secondary Study	Reason for not Including as Primary Study
Aydal et al. (2016)	Lack of information about research method
Baca & Carlsson (2011)	Security requirements is not a main focus
Bellomo & Woody (2012)	Security requirements is not a main focus; Some information on research method is lacking
Fitzgerald et al. (2013)	Limited focus on security requirements work
Ghani et al. (2014)	Lack of information about research method
Kongsli (2006)	Experience report
Nicolaysen et al. (2010) (case study part)	Case study in research project and with unclear research method
Pohl & Hof (2015)	Evaluation with students; weak research method
Rajba (2018)	Lack of information about research method
Renatus et al. (2015)	Security requirements is not a main focus; unclear research method
Rindell et al. (2016)	Security requirements is not a main focus
Savola et al. (2012)	Limited focus on security requirements work
Sachdeva & Chung (2017)	Lack of information about research method
Williams et al. (2009)	Evaluations with students

cover software security in agile development in a broader sense, and then move on to studies of specific techniques relevant for software security work. The study by Terpstra et al. (2017), although it could be considered part of the first study category, is described in a separate section. This is because Terpstra et al. provide a conceptual framework based on their findings, thus concepts from this study have already been identified.

### **Studies That Provide an Overview of Security Requirements Work and Challenges**

Six of the primary studies concern software security in agile development in a general sense, not tied to any particular security technique and not limited to security requirements work. Five of these studies use interviews as the mean of data collection. Adelyar and Norta performed interviews with 3-4 representatives from four teams, with the goal to identify security challenges in agile practices (Adelyar & Norta, 2016). Bartsch performed ten interviews with participants from nine companies, with the goal to “expand on the theoretical findings on security-critical agile development through an exploration of the challenges and their mitigations in typical agile development projects” (Bartsch, 2011). In the interviews, they focused on the following topics: “Customer involvement”, “Developer security awareness and expertise”, “Effects of ‘agile’ on security”, “Security practices” and “Authorization”. van der Heijden et al. performed ten interviews with varying roles in five teams, to understand challenges in large-scale agile development (van der Heijden, Broasca & Serebrenic, 2018). Nicolaysen et al. (2010) performed six interviews with software developers from different companies who were using agile methodologies. The goal of the interviews was to understand whether software security was a specific concern in agile software development. Tøndel et al. (2017) performed interviews with representatives from 23 different public organizations related to their software security practices and challenges. The goal of the study was to understand how current software organizations can work with security in a risk-centric way, and it included both people in development teams and people in information security positions in the organizations. The organizations mainly used some type of agile development practices. The study by Poller et al. is a case study using a broader set of data collection methods (Poller, Kocksch, Türpe, Epp & Kinder-Kurlanda, 2017). Poller et al. followed a product group over 13 months, starting shortly after an external security audit, and they aimed to explore how the development group’s work routines were affected by this external security audit and training.

Table 4 gives an overview of the main concepts identified from these studies. In the following we introduce the main findings from these studies in more detail.

#### *Adelyar and Norta (2016): Challenges With Agile Practices*

Adelyar and Norta identified several agile practices that posed challenges to important security principles. Frequent changes in software, different developer pairs involved, and unclear and inconsistent requirements and priorities from the customer were found to pose challenges on security, because they limited the possibility for having a system-wide view of the software, a simple design and having an ongoing development attention on security. Additionally, it made it challenging to maintain limitation of privileges and separation of privileges.

#### *Bartsch (2011): Effects of Agile Development on Security*

Bartsch found that the individuals and their relationships are highly important when it comes to security, including whether and to what extent security requirements are identified. The role of customers and developers was explored. For customers, Bartsch found that security awareness among customers was heterogenous. Half of the interviewees talked about problems that stemmed from a lack of security awareness with customers. On the other hand, one interviewee explained about a project where “the customer was very security-aware and developed very specific security requirements because the developers were rather unaware” (Bartsch, 2011). The trust relationship between the customer and the development team impacts security. Often, customers can “only state unclear security requirements leading to implicit security requirements” (Bartsch, 2011) and customers may lack the

Table 4. Overview of concepts from the studies of agile software security overall

Concept	Relevant Findings
Priorities (functionality vs. security)	<ul style="list-style-type: none"> <li>• Functional requirements get prioritized over software security (Nicolaysen et al., 2010; Tøndel et al., 2017)</li> </ul>
Project constraints	<ul style="list-style-type: none"> <li>• Management commits to fixed time and budget, resulting in few resources spent on security (van der Heijden et al., 2018)</li> </ul>
Business case for security	<ul style="list-style-type: none"> <li>• As security was not considered a feature, it was not part of feature requests (that is, not part of expected deliveries and current priorities) (Poller et al., 2017)</li> <li>• Security not seen as part of working software – it costs extra time and money without providing functionality (van der Heijden et al., 2018)</li> </ul>
Pressure (time and other tasks)	<ul style="list-style-type: none"> <li>• Short iterations lead to pressure, which can lead to problems integrating security activities (Bartsch, 2011)</li> <li>• Security gets lost in daily work due to time-pressure and other tasks that need to be finished (Poller et al., 2017)</li> </ul>
Customers and customer relations	<ul style="list-style-type: none"> <li>• Customers' security awareness and priorities is heterogeneous and it impacts software security in the projects (Bartsch, 2011; Nicolaysen et al., 2010)</li> <li>• The trust relationship with customers impacts software security, as non-technical customers have a hard time comprehending security in a technical way and often trust developers to just handle this (Bartsch, 2011)</li> <li>• Vendors are trusted to take care of security (Tøndel et al., 2017)</li> <li>• Customers (Bartsch, 2011) and Product Owners (van der Heijden et al., 2018) contribute to security with their domain knowledge, even if their security awareness is low. Close involvement of the customer/Product Owner is thus recommended.</li> <li>• There are "unclear privileges and responsibilities between customers and developers" (Adelyar and Norta, 2016)</li> </ul>
Individuals and their security posture and competence	<ul style="list-style-type: none"> <li>• "The overall security in a project depends on the security expertise of the individuals, either on the customer or developer side" (Bartsch, 2011)</li> <li>• Architects have a potentially important role, but this is dependent on their personal initiative and interest in security. In practice, few software architects seem to have security as a main interest (Tøndel et al., 2017)</li> <li>• Product Owners are "often not aware enough of the added business value for performing certain security actions" (van der Heijden et al., 2018).</li> <li>• Developers lack intrinsic motivation for security (Poller et al., 2017)</li> <li>• (Lack of) software security competence impacts software security (Nicolaysen et al., 2010)</li> <li>• Perceived threats, in particularly related to reputation (Nicolaysen et al., 2010) and concrete threats expressed in monetary terms (Bartsch, 2011), increase security awareness. However, this increased concern for security does not necessarily lead to a commitment to software security (Nicolaysen et al., 2010).</li> <li>• Much of developers' security expertise is self-taught and come from news and blogs. Developers are motivated to learn security due to a feeling of responsibility for the project with a holistic development approach. (Bartsch, 2011)</li> <li>• There are wide variations in security awareness. Training is important (van der Heijden et al., 2018).</li> <li>• Organizations lack a structured approach for software security training (Tøndel et al., 2017; Baca et al., 2015; Terpstra et al., 2017)</li> <li>• There is a high turnover in development teams, particularly due to inclusion of external consultants. These consultants do as they are asked to, thus if they are not asked to consider security they will not pay attention to it (van der Heijden et al., 2018).</li> </ul>
Responsibility	<ul style="list-style-type: none"> <li>• The responsibility for identifying and deciding on security requirements for the development projects seems fragmented - no one fights for software security (Tøndel et al., 2017)</li> <li>• Accountability for security actions is unclear (van der Heijden et al., 2018)</li> </ul>
Preferred security strategy	<ul style="list-style-type: none"> <li>• Other ways to secure the system (e.g. infrastructure security) reduces the perceived need for software security (Nicolaysen et al., 2010).</li> </ul>

continued on following page

Table 4. Continued

Concept	Relevant Findings
Legislation, audit	<ul style="list-style-type: none"> <li>• Legal requirements are a key driver for performing risk analysis (Tøndel et al., 2017)</li> <li>• Privacy legislation can make it difficult to work according to agile principles (Nicolaysen et al., 2010)</li> <li>• External audits can increase security motivation of developers (Bartsch, 2011; Poller et al., 2017)</li> </ul>
Communication	<ul style="list-style-type: none"> <li>• Improved communication among developers and quality assurance impacts security motivation of developers (Bartsch, 2011)</li> <li>• Intra-company competition can impact security motivation of developers (Bartsch, 2011)</li> <li>• Developers may hold incorrect assumptions about managers' security priorities when these are not made explicit (Poller et al., 2017)</li> <li>• Security awareness and expertise spreads between developers in informal discussions (Bartsch, 2011)</li> <li>• Important decisions are made in sprint meetings, and security people are not present in these meetings (Tøndel et al., 2017)</li> <li>• Security people are sometimes involved, but seem to be passive, either waiting to be invited or participating in the beginning and then leaving the project to fend for itself (Tøndel et al., 2017)</li> <li>• Silo structure - security and legal competence in the organizations does not necessarily benefit development (Tøndel et al., 2017)</li> <li>• There is tension between different groups, e.g. between architects and legal/security experts. Hard to make compromises. (Tøndel et al., 2017) There is a lack of understanding between information security officers and the development team; feels like "chasing different goals" (van der Heijden et al., 2018).</li> <li>• Lack of documentation makes communication between the team and the security officer ineffective (van der Heijden et al., 2018)</li> <li>• Security-related information should be easily available to the team (van der Heijden et al., 2018)</li> <li>• Close involvement with a Security Officer is beneficial for teams, especially since this increases acceptance of security (understand why) (van der Heijden et al., 2018)</li> </ul>
Development approach	<ul style="list-style-type: none"> <li>• A holistic development approach can lead to a more complete picture of the system for developers, and can impact developers' sense of responsibility for security. A more complete picture of the system can additionally, together with iterative and incremental development, lead to improved and simpler design (Bartsch, 2011).</li> <li>• Team autonomy can make it more difficult for managers to prescribe security activities (Poller et al., 2017)</li> <li>• Frequent changes in software requirements cause repetition of work, pressure on developers, more complex designs, illogical sequences of integration. This impacts the attention developers give to security, and make it hard to keep a system-wide view and demonstrate that the important threats have been identified and mitigated (Adelvar and Norta, 2016).</li> </ul>
Representation of security requirements	<ul style="list-style-type: none"> <li>• Security requirements can be implicit or explicit. Customers can often only state implicit and unclear security requirements (Bartsch, 2011).</li> <li>• Security was considered a matter of quality, and developers were expected to deal with quality matters without these being explicit and visible (Poller et al., 2017)</li> <li>• Developers often derive security requirements from functional requirements. Some document them as part of Definition of Done (DoD) to make the security requirements explicit (Bartsch, 2011).</li> <li>• Having a formal security requirements process can be considered too theoretical and bureaucratic (Poller et al., 2017)</li> <li>• It is unclear what it means to "properly take care of security concerns", e.g. what the documentation requirements are (van der Heijden et al., 2018)</li> <li>• The security requirements formulated by security management were considered too technical, but also ambiguous. From the security side there is the desire to keep the requirements generic (van der Heijden et al., 2018).</li> </ul>
Iterative process	<ul style="list-style-type: none"> <li>• Security requirements are usually refined over several discussions and iterations. Functional changes as well as the complexity of security requirements can impact the need for refinement and iterations on the security requirements (Bartsch, 2011).</li> </ul>



necessary technical expertise to understand the basis for the security measures. Thus, customers usually assume that developers take appropriate measures to ensure adequate quality. However, a majority of the interviewees mention that “irrespective of the customer’s security awareness, close customer participation improves the security requirements elicitation with their domain knowledge” (Bartsch, 2011). For developers, their individual interest in and sense of responsibility for security is important as security awareness is generally built in an informal way; security knowledge is spread as part of informal discussions and is often self-taught from news sources and blogs.

Bartsch found that the agile development approach has benefits when it comes to security, despite some well-known challenges (e.g. “neglected assurance practices from the pressure of short iterations” (Bartsch, 2011)). Agile practices can bring on a simpler software design and a more holistic development approach for the individual developer. Bartsch found that this could lead developers to feel responsible for the project, and thus increase their motivation regarding security. Compared to pre-agile development, interviewees stated that “improved communication among developers and quality assurance helped” (Bartsch, 2011) in addition to external audits and intra-company competition on quality.

In general, security requirements are refined over several iterations. Bartsch explains that in one project the authorization requirements were complicated and difficult to elicit bottom up, thus a simpler top down model was implemented that then had to be refined and adapted in production. In another project, functional changes repeatedly required security to be discussed.

#### *Van der Heijden et al. (2018): Challenges in Large-Scale Agile Development*

Of the challenges that van der Heijden et al. (2018) identified in their study, they were particularly concerned with which challenges were specific for large-scale agile development. These were “alignment of security in a distributed setting”, “developing a common understanding of roles and responsibilities”, and “integration of low-overhead security testing tools”. In addition, the study identified challenges that had been identified previously in the study performed by Bartsch (2011), and thus was considered to be challenges also in smaller-scale agile: “implementing low-overhead security documentation”, “spreading security awareness and expertise in the team”, “formulating clear security requirements”, and “fostering Product Owner commitment to security”.

#### *Nicolaysen et al. (2010): Software Security as a Concern in Agile Development*

Nicolaysen et al. found that many factors negatively impact how the need for software security is perceived and prioritized. In general functionality is given priority. About half of the customers express some security concerns, but customers’ influence on security is not necessarily positive. They give an example of this; one customer “thwarted a security solution [...] because they did not like it” (Nicolaysen et al., 2010). The studied companies have a lack of security competence, few state that they have experienced any security breaches, and in general security protection is achieved through the infrastructure (e.g. firewalls). Reputation damage is something that worries the interviewees, but the worry is not enough to commit to increased security efforts. Nicolaysen et al. state that that “[n] one of the companies had found or created any fully developed technique for integrating software security into agile software development” (Nicolaysen et al., 2010).

#### *Tøndel et al. (2017): Risk Centric Software Security Practices*

Tøndel et al. found that practices in the studied organizations were not risk based, although the organizations performed some activities that could be said to be part of a risk-based approach to security. Legal requirements were found to be an important driver for software security activities and requirements.

Responsibility for software security was fragmented in many of the studied organizations. In particular it seemed unclear where the responsibilities of security people in the organization end and where the responsibility of the development organization starts when it comes to software security.

Although the organizations might have security experts in-house, this expertise did not necessarily benefit the security work in the development projects, due to the silo structure of the organizations. People working on security or other non-functional requirements did not necessarily have a place at the table when important decisions on security were made in the projects. In many projects, involvement of security expertise was considered challenging because development was done by external vendors. The organizations offered limited formal training on software security. Software architects were pointed out as potential allies in the software security work, but with the challenge that few architects were considered to be particularly interested in security. As a result, it seemed arbitrary whether or not security was considered for the projects. In general, functionality was often prioritized over security.

*Poller et al. (2017): Effects of External Security Audits on Organizational Change in Relation to Software Security*

Poller et al. (2017) found that software security was considered a quality aspect among other quality aspects, and that in the studied company developers were thus expected to deal with security (as with other quality aspects) without this being made explicit and visible. This was considered by Poller et al. as a main reason for software security work not being established in the company. For developers, the feature requests represented expected deliveries, and as security was not considered a feature it was not on the feature request list. This resulted in security being perceived as not important by some developers. There was a perception that managers “would see security as being in a resource conflict with feature development” (Poller et al., 2017). The study however found that managers did not seem opposed to security, but rather that security “had not yet come to their specific attention” (Poller et al., 2017), and that it was considered a quality matter that developers were trusted to deal with as a technical issue. Additionally, security lacked visibility in the team and the developers in general lacked intrinsic motivation for security; their motivation was to “put something together and seeing it work” (Poller et al., 2017).

The study identified challenges with having autonomous and self-organizing teams in that managers had limited means of prescribing security activities. Instead they had to rely on less direct approaches, e.g. indicators or training. Developers, on the other hand, found that security got “lost in the daily work since we always have time-pressure, the release needs to be finished, tests need to be done” (Poller et al., 2017), thus they did not find time to really go deep on security. Additionally, lack of resources was considered one reason.

There was an attempt by security experts to establish a formal security requirements elicitation process, but this met resistance from managers and developers because it was considered theoretical and bureaucratic, and they were not convinced it would improve security.

*Related Findings From Secondary Studies*

Baca and Carlsson (2011) used interviews to evaluate the cost and benefit of the Microsoft SDL, the Cigital Touchpoints and Common Criteria for agile development. They found that none of these approaches were a good match with agile development because of high cost and a lack of benefits. However, the activity of writing security requirements was endorsed, as developers believed it could help identify easy gains and help guide the project. Aydal et al. (2006) demonstrated that software security can be integrated with XP practices. In their study, security requirements were introduced rather late in the process and they found that this could lead to many changes in the system. They suggested using the Planning Game to establish security requirements within iterative and incremental development.

There is some evidence in the study of Savola, Frühwirth & Pietikäinen (2012) that indicate that regulations in a domain can impact the work on security requirements. In their study on metrics they found that “the practitioners emphasized compliance (with legal and industry regulations, customers’ needs and organizational policies), whereas 80% of researchers emphasized the metrics’ ability to

offer a high-level overview of security” (Savola et al., 2012). Rindell, Hyrynsalmi & Leppänen (2016) report on using Scrum for a regulated project with positive results. In the studied project, security was however to some extent viewed as being on the side of the project (implemented in side tracks to the main sprint cycle) and much of the extra security related work was documentation related.

Rajba (2018) presents a journey of one company in improving their security processes. Challenges identified in the beginning of this journey included complex security checklists that were considered too technical and not relevant, people being more interested in passing security reviews than making more secure applications, challenges in scoping the security work so as to not having to undertake too much at once, repetitive tasks, and a lack of documentation, security requirements and knowledge. Many of these challenges were addressed with improvements in training and security checklists, provision of templates, tool support, and improved use of an internal security review team.

Nicolaysen et al. (2010), in addition to reporting on interviews (as described above), report on a case study of a research project. Due to the domain (healthcare), security was initially given attention in the studied project. In the end, however, the resulting product had many security concerns (vulnerable to seven out of the OWASP top 10 issues), and they found that “only the functional results of the security design made it out of the backlog [...] leaving most non-functional security aspects alone in the dark” (Nicolaysen et al., 2010). Nicolaysen et al. point to some reasons for this, mainly a lack of continuity in the security experts assigned to the project, resulting in delays. Thus, the security design was not completed as planned and implementation started before security had been properly considered. Communication problems is also mentioned, although Nicolaysen et al. is not concrete on what kind of communication problems there were and how they influenced development.

### Studies of Specific Techniques

Though there are several suggested techniques for identifying and working with security requirements in agile development, few of these techniques have been studied and evaluated in an industrial environment. Two of the primary studies we identified study specific techniques related to software security requirements work. Baca et al. performed an action research study at Ericsson, where they studied the effects of implementing a security-enhanced agile software development process (SEAP) (Baca, Boldt, Carlsson & Jacobsson, 2015). This process includes several software security activities (e.g. code review, penetration testing), but the study reported focused on two key aspects of SEAP: adding more security resources in the project and the development teams, and performing incremental risk analysis. The study of SEAP included one product with 88 staff members distributed among 8 development teams. Four versions of the product were considered, of which the three latest versions were developed using SEAP. Effort and identification and treatment of risk was compared between versions.

Williams et al. proposed Protection Poker (Williams, Meneely & Gegick, 2010), which is a technique for security risk estimation of requirements, as well as for security awareness building and exchange of security information in the team. The technique is particularly suited for agile teams, and Protection Poker is intended to be played in the planning meeting of every development iteration. The effects of using Protection Poker were studied in a case study including one maintenance team at Red Hat. The team had eleven participants (seven developers, three testers and one manager), and used Scrum as their development methodology. The team studied had no security expert, and the knowledge of software security varied among team members; some very knowledgeable, some relatively new to software security. The study lasted for three months with five Protection Poker sessions in total. Data collection was done using observation and a short survey.

Neither SEAP nor Protection Poker is specifically about security requirements. However, performing incremental risk analysis and doing security risk estimation could be considered part of security requirements work as defined in this paper. Table 5 gives an overview of the main concepts identified from these two studies. In addition to the findings from the studies of the techniques themselves that are used as a basis for this table, an underlying assumption is that the technique



Table 5. Overview of concepts from the studies of security requirements techniques

Concept	Relevant Findings
Incremental security analysis in the team	<ul style="list-style-type: none"> <li>• An incremental risk analysis process improves identification and handling of risk. Security issues are solved in the team (distributed, not centralized), more detailed analysis is performed, more severe risks are identified and more risks are corrected. This leads to more cost-effective risk management. (Baca et al., 2015)</li> </ul>
Security resources in team	<ul style="list-style-type: none"> <li>• With security resources in the project it is possible to work distributed and solve issues in the team (Baca et al., 2015).</li> </ul>
Security discussions in the team	<ul style="list-style-type: none"> <li>• Using a technique for discussing security implications of functional requirements in the full team leads to improvements in software security as it results in improved spread of security knowledge and improved security skills (e.g. skills to think like an attacker), in addition to leading to identification of security needs in the project in form of security requirements, training needs and security activities (Williams et al., 2009).</li> </ul>



itself is a factor that impacts security requirements work. In the following we explain the results of the two studies in more detail.

**Baca et al. (2015): The Effect of Added Security Resources and a Distributed and Incremental Approach to Risk Assessment in an Agile Development Project**

In their study, the introduction of SEAP was found to improve identification and handling of risk, and because of this, the risk management was found to be more cost-efficient than with the approach previously used by Ericsson.

Three aspects are however important to note related to this study. First, the process for risk analysis used is not explained in detail. In the study, the risk analysis of four software versions of the same product is compared, where v1.0 was developed using the traditional Ericsson approach, and v2.0 - 4.0 were developed using SEAP. With the traditional approach, risk analysis was performed once a year (per release) and involved six to eight persons for a day. With SEAP, the frequency of risk analysis was 30-40 per year, involving three to four persons for an hour each time. The scope of risk analysis with SEAP is much smaller than with the traditional approach. Another main difference between SEAP and the traditional approach is the security resources involved in the analysis and in the project in general. Traditionally the risk analysis was led by the security manager, and this role was not directly involved in the development. With SEAP, more security resources are added to the project (the equivalent of four full time positions), and one of these roles (the security master) is assigned to two or three teams (25% per team). Available time apart from security work is spent as a regular developer. The security master leads the risk analysis work. Based on the description of the risk assessment process in SEAP we thus know that the frequency is increased, the scope for each analysis is reduced, and the approach is more distributed.

Second, the reason for the identified improvement is not discussed in detail in the paper. The authors claim that a main reason for the improvement is that security issues are dealt with in a more distributed fashion, and thus more issues are solved directly by the team. Though not discussed in the paper, it should also be expected that when security resources are added to the team, the security people are more likely to understand the product and thus their analysis is likely to improve. However, there are alternative explanations that are not discussed by the authors. One example of a factor that may have influenced the results is related to the study design and its use of comparison. The traditional approach was used for v1.0, and SEAP for later versions. The ability to identify more high risks with SEAP may be due to the method, but may also be because v2.0 and up contain more risky features. This, and other alternative explanations, are not discussed by the authors.

Third, the product developed in the study was related to online money transfer and was thus considered to be security-critical. This allowed investing in the additional security resources required



by SEAP. Thus, we do not know whether or not the resources needed for SEAP can be justified for projects that are less security-critical.

#### *Williams et al. (2010): Risk Estimation Using Protection Poker*

Williams et al. found that a main effect of using Protection Poker in this case study was that software security knowledge was spread among team members, and key risks were discussed; “[d]uring Protection Poker sessions, all team members participated in the conversation - some asking questions, some sharing their software security expertise, all becoming incrementally better at thinking like an attacker with each Protection Poker session” (Williams et al., 2010). The playing of Protection Poker led to revisions of two requirements for added security, resulted in a request for education on cross-site scripting, identified the need for more security testing, etc. It was found that Protection Poker supported participation from all team members, also those with passive, quiet personalities. Note however, that we cannot say from this study whether similar results could have been achieved with another technique.

#### *Related Findings From Secondary Studies*

Protection Poker has also been studied in a trial with 50 advanced undergraduate students taking a software engineering course (Williams, Gegick & Meneely, 2009). In that study, it was found that Protection Poker resulted in more discussion and learning about software security compared to previous semesters where Protection Poker was not used. In the discussions, general lessons about security surfaced fast, e.g. discussions on input validations, common exploits, etc.

Kongsli (2006) reports on experiences with using misuse stories and automatic testing of security in the development of web applications, and points to similar benefits as Baca et al. (2015) and Williams et al. (2010) although using a different technique (misuse stories). Reported benefits include increased security awareness in the development team and team ownership of security issues. Additionally, Kongsli reports that when security is sufficiently broken down (in terms of misuse stories) it is easier to relate to and handle by developers, though with the risk of misuse story incompleteness as security concerns not directly related to a user story can be overlooked. This risk is not pointed out by Baca et al. (2015) and Williams et al. (2010). Kongsli also points out that the need for a security specialist on the team is not completely eliminated with the used security techniques.

Ghani, Azham and Jeong (2014) suggest adding the Security Backlog and the role of a Security Master to Scrum, and evaluate how this impacts agility. Results are positive, in that agility is actually found to slightly improve. This may be due to adding more security expertise and workforce.

Renatus, Teichmann and Eichler (2015) suggested a method for threat assessment in line with agile principles and a method for evaluating agility of methods, and the method itself was evaluated in one SME. Central to the method they suggested was a split between the tasks of developers without in-depth security expertise and the security curator. Security curators got the task of pre-modeling the features soon to be implemented, while the developers were tasked with figuring out how to implement the controls (Renatus et al., 2015). This is in line with SEAP and Protection Poker when it comes to an incremental approach to development but represents a slightly different approach to dealing with the need for security expertise. Renatus et al. report that it was seen as a valuable approach by the SME.

The positive effect of incremental security analysis is supported by findings from Fitzgerald, O’Sullivan & O’Brien (2013) who found that continuous compliance activities and transparency of project status facilitate risk mitigation. In particular they point to benefits of risk prioritization, as tackling the most significant risks first can improve risk mitigation. Bellomo & Woody (2012) report on an interview study among agile program managers and Accreditation Reviewers at the Department of Defense (DoD), mainly concerning high risk software where accreditation is necessary. Bellomo and Woody underline the importance to support prioritisation of security requirements and the need for security expertise being available to the team. Additionally, they advocate a risk-based incremental approach to security feature design and development, as this can mitigate the temptation to “focus on

delivering the low hanging fruits first (the easy stuff) and ignore developing the more complex, high risk capabilities” (Bellomo & Woody, 2012). Bellomo and Woody additionally found that enforced use of a security impact assessment field in the backlog increases the likelihood that security risks are continuously assessed.

Sachdeva & Chung (2017) report on a case study of two software projects, one in which security and performance requirements were handled implicitly and as an afterthought, and one in which they were identified and addressed early and added to the backlog. They found clear benefits of the latter approach. Thus, though other studies point to benefits of incremental analysis, this study point to the importance of early inclusion of security.

Pohl and Hof (2015) suggested Secure Scrum; a way to integrate security into development without changing the underlying Scrum process. Their evaluation of Secure Scrum comes with its weaknesses; relying on small student projects that lasted only a week. Bearing this in mind, the results from this study are relevant as they point to effects of having a security technique. Pohl and Hof found that security was not taken care of by the student developers that did not use Secure Scrum, but that when equipped with this technique they were able to elicit security requirements and implement some of these requirements. Security techniques can thus act as the reminder that is needed by developers to include software security.

### Conceptual Framework by Terpstra et al. (2017)

Terpstra et al. (2017) created a conceptual framework based on the results they got in their study of professionals’ posts on LinkedIn. This conceptual framework is shown in Figure 1. In the following we explain the conceptual categories of Terpstra et al. in more detail.

The conceptual category *perceptions of priority* was used by Terpstra et al. to represent issues related to prioritization of security requirements at inter-iteration time. They found that customers and business representatives often push for functionality and do not prioritize security. However, developers’ priorities may be different. This is related to the conceptual category *ownership of security requirements* that was used by Terpstra et al. to represent findings that show that no role takes or is given full responsibility for security requirements in development projects. As stated by Terpstra et al. “business representatives and product owners usually have little awareness of security requirements and rarely work towards their elaboration early on” (Terpstra et al., 2017). This is supported by identified challenges such as “[t]he product owner has often too much power and instills his attitude of treating non-functional requirements”, and “[t]he product owner is sometimes acting like a business owner or stakeholder and pushes only for features” (Terpstra et al., 2017). Additionally, they found that “developers who understand risks associated with poorly treated security requirements, may not know how to communicate the possible security issues to their product owner and convincingly present him with information on how much it would cost if not fixed and if a problem arises” (Terpstra et al., 2017).

These challenges can in part be explained by findings related to the conceptual category *business case*. Some of the challenges identified by Terpstra et al. was that “[a]gile techniques are business-value driven” and “[s]ecurity is hard to ‘sell’ as a business value”. Additionally, “[s]ecurity requirements cost money to elaborate due to experts’ involvement” and “[p]eople drop security because they perceive it a fight not worth fighting” (Terpstra et al., 2017). To add to this, the conceptual category *attitude towards security requirements* was used by Terpstra et al. to represent findings that in some cases “team members ‘do not care’ about security requirements just because there is no incentive to do so (...). Or, because no one really understands completely what these requirements are” (Terpstra et al., 2017). Terpstra et al. found that using security regulation to justify the security requirements was one coping strategy used by practitioners.

The conceptual category *organizational setup* was used by Terpstra et al. to represent findings that show that the organizational culture can both help and hurt the security requirements work. Terpstra et al. in particular found that the organizations’ approaches to educating developers on software security could have an impact. Coping strategies identified include educating the business on security, raising

awareness in the development team, adding a security expert to the team, making sure the product owner is supporting security, and having cross functional streams to help not forgetting about security. Some of the challenges identified by Terpstra further explain how the agile development approach in itself may be a challenge in having security being prioritized, e.g.: “People do care about security, but do not think about it”; “Agile techniques are vulnerable for forgetting things like security” and; “Security requirements get often delivered in the last minute” (Terpstra et al., 2017). The conceptual category *definition of “done” (DoD)* was used by Terpstra et al. to represent the opinion of some of the professionals that the DoD should include security requirements. They found that security requirements often were poorly defined, and that coping strategies included integrating security into the DoD, estimates, acceptance criteria and user stories.

## RESULTS: THE MAIN CONCEPTS AND THEIR RELATIONS

In the previous section we identified several concepts relevant to security requirements work as reported in the identified primary studies (see Table 3 and Table 4). Based on the concepts we identified, as well as those identified by Terpstra et al. (2017), we then identified what we consider the most important and prevalent concepts in the primary studies, and the relations between these concepts. We used this to create a conceptual framework with a graphical representation. In this section we describe the result of this work.

### Main Concepts

Table 6 shows how the concepts from the primary papers have been grouped into a set of main concepts. The main concepts are as follows:

- **Teams’ security posture and competence:** The security awareness and competence of the team and the individual team members are important in remembering security, identifying the need for security, following it up with performing security activities and in having the competence needed to adequately handling the security (Terpstra et al., 2017; Bartsch, 2011; Nicolaysen et al., 2010). Benefits have been identified that can be tied to a decentralized approach to security analysis (Baca et al., 2015), but this implies commitment and capability of the development teams in doing this work;
- **Customers’ security posture and competence:** The interviews reported by Bartsch (2011) in particular, but also the interviews reported by Nicolaysen et al. (2010), show the importance the customer plays in the work on security requirements. Both these studies show that customers have the influence both to drive and hinder the work on security requirements;
- **Customer relation and involvement:** Customers have been found to provide valuable competence to the discussions on security requirements, and their competence and the trust relationship with the developers can influence how security requirements are specified (Bartsch, 2011; van der Heijden et al., 2018);
- **Business case for security:** Functionality is often prioritized over security (Nicolaysen et al., 2010; Terpstra et al., 2017). Security is in many cases not seen as part of the software or something that adds value, but rather as a cost (Terpstra et al., 2017; Poller et al., 2017; van der Heijden et al., 2018). However, legislation or audits that put requirements on security can motivate security effort (Terpstra et al., 2017; Nicolaysen et al., 2010; Bartsch, 2011);
- **Organizational culture and setup:** Several aspects with the organizational culture have been found to have an effect on security requirements work. Examples are the communication between teams and central resources on quality (Bartsch, 2011) and the organization’s approach to software security training (Terpstra et al., 2017). In addition, the organization has the potential to make decisions that impact what security resources are available in a team and the formal ownership for software security in projects;

Table 6. Main concepts

Main Concepts	Table 4 Concepts	Table 5 Concepts	Concepts Terpstra et al. (2017)
Teams' security posture and abilities	Individuals and their security posture and competence; Responsibility; Preferred security strategy	Security resources in the team	Attitude towards security
Customers' security posture and competence	Customers and customer relations; Individuals and their security posture and competence; Preferred security strategy	-	Perceptions of priority
Customer relation and involvement	Customers and customer relations	-	-
Business case for security	Priorities; Project constraints; Business case for security; Pressure; Legislation, audit	-	Business case
Organizational culture and setup	Communication; Development approach; Responsibility	Security resources in the team	Organizational setup; Ownership of security requirements
Process for making priorities on requirements	Priorities; Project constraints; Representation of security requirements	-	Perceptions of priority
Development approach	Development approach; Pressure	-	-
Security requirements elicitation approach	Iterative process,	Incremental security analysis in the team; Security discussions in the team	-
Security requirements representation	Representation of security requirements	-	Definition of "done"

- **Process for making priorities on requirements:** In agile projects, decisions on what security work to prioritize can be made without any security experts being involved and the decision can be highly dependent on the security posture of individuals (Terpstra et al., 2017);
- **Development approach:** An agile development approach can impact the security requirements positively, e.g. it has been found that developers with a holistic view of the software they develop can feel more responsibility for security (Bartsch, 2011). However, there are also known challenges (e.g. pressure of short iterations) (Bartsch, 2011; Nicolaysen et al., 2010) and frequent changes (Adelyar & Norta, 2016);
- **Security requirements elicitation approach:** Having a defined process for security to make sure security is remembered throughout the project can make a difference. Additionally, the approach selected can impact the effect of the work. As an example, approaches such as Protection Poker where the full team discusses security can lead to certain effects that would maybe not be present in a more expert oriented approach. Security requirements work is commonly considered to be iterative (Bartsch, 2011) and this should be supported by any selected elicitation approach;
- **Security requirements representation:** Security requirements often end up being implicit (Bartsch, 2011; Adelyar & Norta, 2016). Having security requirements as part of the Definition of Done is one suggested way to make them more explicit and actionable (Bartsch, 2011; Terpstra et al., 2017).



## Identifying Relations Between Concepts

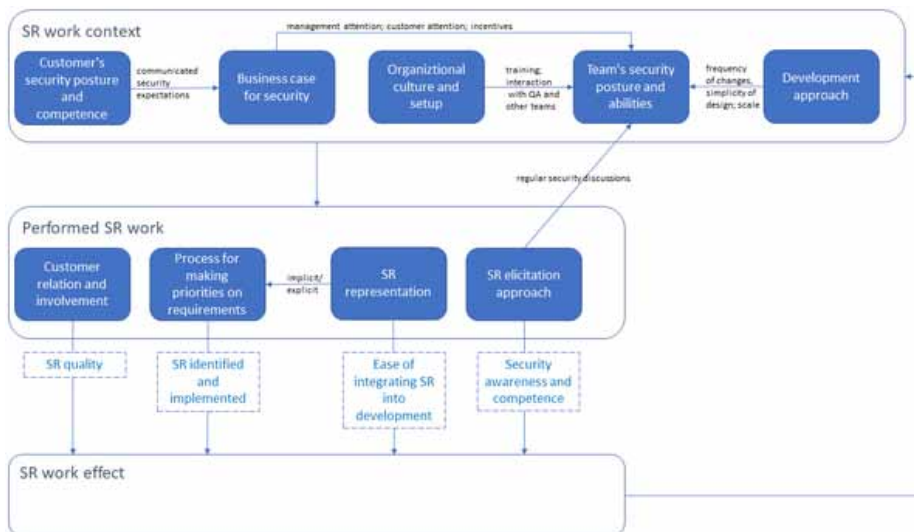
In the introduction, security requirements work was described as comprising activities to: 1) decide whether and how to identify security needs, risks or requirements for a project; 2) do the requirements elicitation; 3) communicate the identified security needs, risks or requirements, and; 4) integrate these and make priorities related to them in development. These security requirements work activities take part in a context that highly influence this work in various ways. Figure 4 depicts the conceptual framework we ended up with based on the analysis of the selected papers. Here we have divided the identified concepts into two main categories: 1) contextual factors, i.e. factors that are outside the requirements work itself, but impact the security requirements work in some way (e.g. impact the priority the security work is given, who participates, how it is done, etc.), and; 2) concepts related to the more practical aspects of the work and how it is performed (e.g. who actually participates in an activity, how the work is actually done, etc.).

In the following we describe the relations between the concepts in more detail. Additionally, we introduce a third overall category that is largely missing from the identified papers, namely that of the effect of the security requirements work.

### Security Requirements Work Context

The team's security posture and abilities can be influenced by a number of factors. An obvious influence is training (Bartsch, 2011; Terpstra et al., 2017), however, this training does not need to be formal. Protection Poker is an example of a technique that has been found to spread security awareness and knowledge in a team through regular security discussions. Additionally, teams can increase their security competence through communication with quality assurance functions in the company or even a sense of competition with other teams (Bartsch, 2011). Aspects of the development work can additionally have a major impact on security posture of the team. The size and scale of the project itself can impact what type of challenges a project experience in their security work (van der Heijden et al., 2018). Adelyar & Norta (2016) found that frequent changes to software under time

Figure 4. Conceptual framework based on the selected empirical studies (SR is in the figure an abbreviation of 'security requirements')



pressure negatively impacted developers' security attention. Additionally, they found that it impacted the ability to have a simple design and made the software more complex. Bartsch (2011) found that having a holistic development approach can motivate software security and lead to simpler designs. Agile development methods can thus impact positively or negatively on the security posture and abilities of the team depending on the circumstances.

Customers' security posture and competence and the way customers are involved in the security requirements work impact software security work in many ways. It can, together with the relation between the team and customer, impact how security requirements are initially presented, especially their quality and whether they are implicit or explicit (Bartsch, 2011). Additionally, customers' security posture impacts the business case for security, e.g. by the customer making security a clear priority (Bartsch, 2011; Nicolaysen et al., 2010; Terpstra et al., 2017). The business case again influences the security posture of the individuals involved (Terpstra et al., 2017).

### *Performed Security Requirements Work*

Several of the concepts identified fall within the category of performed security requirements work. The relations between these concepts (e.g. how the way requirements are elicited influence how they are prioritized, etc.) are however not discussed much in the papers we have studied. The main relation present is that of the impact of having implicit vs. explicit security requirements (Bartsch, 2011; Poller et al., 2017), and having security included as a feature (Poller et al., 2017).

### *Security Requirements Work Effect*

We find that one category is largely missing from the primary and secondary studies we have identified, namely that of the effect; what makes the security requirements work useful in terms of impact. Figure 4, that shows the conceptual framework we ended up with based on the primary studies, thus includes this effect but without further concepts to help understand it. To move towards an understanding of the effect, we have however added what we understand from the sources to be potential effects of the factors included in the category 'Performed security requirements work', namely the quality of security requirements, the fact that they are identified and implemented, how easy they are to integrate into development, and the security awareness and competence that is built by doing security requirements work. Though we do not have any solid evidence to support that these are important factors characterizing the effect of security requirements work, these have support in the identified papers and can point towards factors that potentially are important for the effect of this type of work.

One effect of software security work that is somewhat available in the primary studies is that of cost. We have however not added that effect to the conceptual framework as it is not clear from the sources what the cost-benefit relationship associated with the security requirements work part is. Cost is however one likely factor of the effect of software security requirements work as well.

Figure 4 shows a possible relationship between the effect of the security requirements work and the security requirements work context. In the papers we build on, there are some pointers to the potential of security requirements work to impact the context, e.g. in form of changes in security competence and awareness among key actors, such as the Product Owner.

## **DISCUSSION**

In the following we discuss recommendations for future research based on the conceptual framework we developed, followed by a discussion of the validity of the conceptual framework.

### **Implications for Research**

The conceptual framework depicted in Figure 4 shows that several contextual factors influence the software security requirements work in agile development projects. This can be understood in more than one way. One possible understanding is that the contextual factors are the factors that are best

understood in the underlying research, and thus future research should aim to identify and understand also factors related to the security requirements work itself and the outcome. However, another possible understanding is that contextual factors are highly important for security requirements work and thus need to be properly understood in order to have an effective approach to software security requirements work in an agile setting. This may point to the need for more research on these factors, also keeping in mind that the contextual factors included in Figure 4 are rather complex, covering characteristics of individuals, the organization, and their interactions.

In the empirical studies we identified, the role of the approach or technique used for security requirements work is not clearly understood. Though there have been studies of different techniques and approaches, there is a difference between evaluating one technique and finding out the effects of that technique vs. understanding what makes the technique behave as it does compared to other techniques. We believe there is a need for more studies evaluating various techniques and approaches, especially in industry settings and over longer periods of time.

From the conceptual framework in Figure 4 one can see that the empirical studies we used provide limited understanding of what causes security requirements work to have effect. It may be more difficult to study and understand the effect of the work than to understand factors impacting the security requirements work, since effects may be longer term and harder to pinpoint. Still, the motivation of doing security requirements work would be an adequate level of implemented security, and if a security requirements approach does not make a significant contribution towards that then it is not worth the effort.

As can be seen from Table 7, many of the concepts we identified can be said to be directly related to the values of the Agile Manifesto (Beck et al., 2001); “[i]ndividuals and interactions over processes and tools” (V1 in Table 7), “[w]orking software over comprehensive documentation” (V2), “[c]ustomer collaboration over contract negotiation” (V3), and “[r]esponding to change over following a plan” (V4) (Beck et al., 2001). This points to the conceptual framework being related to agile development in particular, and not to other types of development approaches. However, this is not necessarily the case. Kanniah & Mahrin (2016) identified a set of factors impacting successful implementation of software development practices based on an SLR. Many of the factors they identified are related to the factors we have identified for security requirements work in agile development; the institutional context, people and action, the project context and the system development process are all represented in the conceptual framework in Figure 4. The factors identified by Kanniah and Mahrin are not considered to be specific for agile development. Based on the current evidence it is thus difficult

**Table 7. Main concepts and their relation to the principles of the Agile Manifesto (Beck et al., 2001)**

Identified Main Concept	V1 Individuals	V2 Software	V3 Customer	V4 Change
Teams' security posture and abilities	x			
Customers' security posture and competence	x		x	
Customer relation and involvement	x		x	
Business case for security		x		
Organizational culture and setup	x			
Process for making priorities on requirements	x			x
Development approach	x	x		
Security requirements elicitation approach	x			x
Security requirements representation	x			



to say what impact an agile development approach has on software security work, and thus what in the conceptual framework we developed are specific to agile development, even if the conceptual framework is entirely based on studies done on projects and companies using some kind of agile development approach. Of the four agile principles, it is the value “Individuals and interactions over processes and tools” (Beck et al., 2001) that seems to be most influential on security requirements work. Of the ten factors that Kaniah and Mahrin later identified as the most influential (Kanniah & Mahrin, 2018), a majority can also be considered to be concerned with individuals and interactions.

To sum up, there is a need for a deeper understanding of software security work in agile development. Especially there is a need to understand better what factors are important for the effect of the work, and to understand the role of the particular approach in bringing about this effect. However, a conceptual framework has a role not only in directing research priorities but also to “identify potential validity threats to your conclusions” (Maxwell, 2013). It is clear that contextual factors are important and influence software security work in agile development in many ways, especially factors concerning individuals and their interactions. Thus, understanding these and taking these factors into account in future studies is essential in order to properly understand the findings. Thus, the conceptual framework can be used to guide future research priorities, but also be input to planning and analysis of future empirical studies.

### Validity of the Conceptual Framework

The conceptual framework presented in this paper is based on nine empirical studies; five interview studies addressing software security in agile (Adeyar and Norta, 2016; Bartsch, 2011; Nicolaysen et al., 2010; Tøndel et al., 2017; van der Heijden, 2018), one case study on the impact of external security audits on development (Poller et al., 2017), two evaluations of approaches or techniques relevant for security requirements work (Baca et al., 2015; Williams et al., 2010) and one analysis of professionals’ postings on LinkedIn related to security requirements in agile (Terpstra et al., 2017). These nine studies together address the topic of security requirements work in agile from varying perspectives and using varying methods, something that can be considered a strength. Still, the nine primary studies we build on can be considered to be rather few, thus we have used a set of secondary studies to improve understanding of the concepts identified from the primary studies.

As previously explained, we decided not to do a comprehensive and systematic search for literature, as one would expect if doing an SLR. We made the initial assessment that given the recent SLRs on software security in agile (Khaim et al., 2016; Oueslati et al., 2015) that we could use as a basis for this work, it was not worthwhile to do a comprehensive search for literature. At later stages in the process we used even more recent SLRs (Alsaquaf et al., 2017; Bishop & Rowland, 2019; Saldanha & Zorzo, 2018; Villamizar et al., 2018) to add to the initial selection of papers. Deciding not to do a full SLR is a weakness of our approach, and it can potentially have impacted the conceptual framework we ended up with, as more identified studies could have resulted in more and/or different concepts and relations between them. However, we never intended this conceptual framework to be a complete and “finalised” conceptual framework, but rather a work in progress that should be improved as more research becomes available (Maxwell, 2013). We would additionally like to point out that the SLRs we used to identify papers seem to vary in what papers are included (see Table 1), something that may indicate challenges in identifying all relevant papers also when doing an SLR. Several of the SLRs we have used as a basis state that the current number of published empirical studies on software security in agile development is rather low (Alsaquaf et al., 2017; Bishop & Rowland, 2019; Saldanha & Zorzo, 2018; Villamizar et al., 2018), thus identification of a high number of studies should not be expected regardless of method for identifying studies.

The conceptual framework presented in this paper is based solely on published empirical studies. Restricting ourselves to only using such studies as a basis for the conceptual framework represents a narrowing of focus, ignoring other sources of knowledge of security requirements work such as unpublished results and the general experiences of researchers and practitioners (Maxwell, 2013;



Robson, 2011). Also, for this reason, this conceptual framework is to be considered work in progress, and something that will need to be refined including more sources.

Relying on published empirical studies additionally pose limitations in that we only have access to as much information about the studies as is available in the published papers. For the study category consisting of more general studies, we would generally have benefited from more information on study context as this would help us understand the results and the selected concepts in more depth. For the studies of specific techniques, the results and thus the concepts are highly related to the specifics of the techniques; if other techniques had been studied it is likely that other concepts would have emerged from the results. It is hard to know what is the effect of the studied approaches (SEAP, Protection Poker) compared to that of other techniques, i.e., which effects are due to the particular way of working in the technique, and which are due to other factors.

Although the work of creating this conceptual framework has been done in a structured way, there is always an element of creativity also in scientific work (Collins, 2019). In this work, the coding of results from the primary sources (step 3), the reorganizing of these codes into themes/concepts for the conceptual framework (step 4) and the development of the graphical representation of the conceptual framework (step 5) all represent some form of creative work, although based on a structured process and although we have aimed to preserve the link between the resulting framework and the findings in the primary sources. It is likely that other researchers would make slightly different categorizations and end up with a different graphical representation of the conceptual framework. In many cases the concepts we ended up with using, both the initial concepts identified based on the primary studies (Table 4 and Table 5) and the main concepts used in the conceptual framework (Table 6), are somewhat overlapping. To illustrate, the security discussions in the teams that are part of Protection Poker in many ways represent one form of incremental security analysis done by the team, and such a discussion influences the security resources in the team. The concepts we ended up with using represent our best effort to group key findings from the primary studies into meaningful and related concepts. The concepts and the framework we ended up with should however not be viewed as a final version, but as a starting point and something that can be improved upon as more empirical research becomes available.

Further evaluation of this conceptual framework is needed. In its current state, the conceptual framework has been primarily developed by one researcher. Future work includes discussing the conceptual framework with more colleagues and validating and improving the conceptual framework when new evidence becomes available. Note however that the concepts in the framework bear similarities to the categories of challenges identified by Oueslati et al. (2015), especially to their categories “[a]wareness and collaboration challenges” and “[s]ecurity management challenges” and to challenges identified by Khaim et al. (2016) and Alsaquaf et al. (2017). Thus, the factors we have identified have been pointed out also by other researchers aiming to understand challenges relating to software security in agile development or quality requirements in agile development. Compared to the conceptual framework developed by Daneva and Wang (2018), it integrates their key concepts of activities, competencies, roles and artefacts, although in a slightly different way. Also, note that the last iteration of the framework that included adding two more primary studies (Adelyar and Norta, 2016; van der Heijden et al., 2018) resulted in only minor updates to the final concepts and to the conceptual framework. Thus, we have reason to believe that this conceptual framework is able to cover the key findings in current empirical research on software security requirements work in agile development.

In the Research Method section, we restated the recommendation from Miles and Huberman (1994) to avoid a non-risk framework with only global level variables and two-directional arrows. The conceptual framework we have presented in this paper is not a non-risk framework, but could be said to be a low-risk framework with many high level concepts and mainly high-level relations between the concepts. This is in many ways a result of limited studies to use as a basis for the conceptual framework. Both Maxwell (2013) and Robson (2011) recommend an inclusive approach at the initial stage. However, the conceptual framework should become more focused as it is refined (Maxwell,

D

2013). Thus, future revisions should move towards more specific concepts, and even excluding concepts that are less important. Revisions can be made based on new data becoming available, or could utilize other sources like experience, a broader set of literature, and thought experiments (Maxwell, 2013; Robson, 2011).

## CONCLUSION AND FUTURE WORK

This paper suggests a conceptual framework for software security requirements work in agile development, with the motivation to increase understanding of this type of work and guide further research. The conceptual framework is based on published empirical studies covering aspects of software security requirements work in agile in an industrial setting. The results point to a need for further empirical studies in this area, especially to improve understanding of factors important for gaining impact from the work on software security requirements in agile projects, as this is largely missing in current work. There is additionally a need for understanding to what extent the concrete approach adopted for security requirements work shape the impact of this work given varying contexts. This would help practitioners in deciding what methods to adopt for their particular case. Contextual factors seem to be highly influential on the way security requirements are treated in current software projects. Thus, these are important to take properly into account in future empirical research studies, especially in plans for data collection and in the analysis phase.

In our own work, we are in the process of using this conceptual framework as an input to planning and analysis of ongoing case studies related to software security requirements work in agile software development. Especially, we plan to use the conceptual framework to provide some structure to the analysis. Additionally, we plan to use the results of the ongoing and future case studies to improve this conceptual framework.

## ACKNOWLEDGMENT

This work was supported by the SoS-Agile: Science of Security in Agile Software Development project, funded by the Research Council of Norway (grant number 247678).

## REFERENCES

- Adelyar, S. H., & Norta, A. (2016, September). Towards a secure agile software development process. In *Proceedings of the 2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)* (pp. 101-106). IEEE. doi:10.1109/QUATIC.2016.028
- Alsaqaf, W., Daneva, M., & Wieringa, R. (2017). Quality requirements in large-scale distributed agile projects—a systematic literature review. In *Proceedings of the International working conference on requirements engineering: Foundation for software quality* (pp. 219–234). Academic Press. doi:10.1007/978-3-319-54045-0\_17
- Aydal, E. G., Paige, R. F., Chivers, H., & Brooke, P. J. (2006). Security planning and refactoring in extreme programming. In *Proceedings of the International conference on extreme programming and agile processes in software engineering* (pp. 154–163). Academic Press. doi:10.1007/11774129\_16
- Baca, D., Boldt, M., Carlsson, B., & Jacobsson, A. (2015). A novel security-enhanced agile software development process applied in an industrial setting. In *Proceedings of the 10th international conference on availability, reliability and security (ARES)* (pp. 11–19). Academic Press. doi:10.1109/ARES.2015.45
- Baca, D., & Carlsson, B. (2011). Agile development with security engineering activities. In *Proceedings of the 2011 international conference on software and systems process* (pp. 149–158). Academic Press.
- Bartsch, S. (2011, Aug). Practitioners’ perspectives on security in agile development. In *Proceedings of the 2011 sixth international conference on Availability, reliability and security (ARES)* (p. 479-484). doi:10.1109/ARES.2011.82
- Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . others (2001). Manifesto for agile software development. Retrieved from <http://www.agilemanifesto.org>
- Bellomo, S., & Woody, C. (2012). DoD Information Assurance and Agile: Challenges and Recommendations Gathered Through Interviews with Agile Program Managers and DoD Accreditation Reviewers. Carnegie-Melon University.
- Beznosov, K. (2003, October). Extreme security engineering: On employing XP practices to achieve ‘good enough security’ without defining it. In *Proceedings of the First ACM Workshop on Business Driven Security Engineering (BizSec)*. Academic Press.
- Bishop, D., & Rowland, P. (2019). Agile and secure software development: An unfinished story. *Issues in Information Systems*, 20(1).
- Collins, H. (2019). *Forms of Life: The Method and Meaning of Sociology*. MIT Press.
- Daneva, M., & Wang, C. (2018, August). Security requirements engineering in the agile era: How does it work in practice? In *Proceedings of the 2018 IEEE 1st International Workshop on Quality Requirements in Agile Projects (QuaRAP)* (pp. 10-13). IEEE.
- Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, 119, 87–108. doi:10.1016/j.jss.2016.06.013
- Fitzgerald, B., Stol, K.-J., O’Sullivan, R., & O’Brien, D. (2013). Scaling agile methods to regulated environments: An industry case study. In *Proceedings of the 2013 international conference on software engineering* (pp. 863–872). Academic Press. doi:10.1109/ICSE.2013.6606635
- Ghani, I., Azham, Z., & Jeong, S. R. (2014). Integrating Software Security into Agile-Scrum Method. *TIIS*, 8(2), 646–663. doi:10.3837/tiis.2014.02.019
- Hanssen, G. K., Stålhane, T., & Myklebust, T. (2018). SafescrumOR -agile development of safety-critical software. Springer.
- Heeager, L. T., & Nielsen, P. A. (2018). A conceptual model of agile software development in a safety-critical context: A systematic literature review. *Information and Software Technology*, 103, 22–39. doi:10.1016/j.infsof.2018.06.004
- Howard, M., & Lipner, S. (2006). *The security development lifecycle*. Microsoft Press.

- Jabareen, Y. (2009). Building a conceptual framework: Philosophy, definitions, and procedure. *International Journal of Qualitative Methods*, 8(4), 49–62. doi:10.1177/160940690900800406
- Kanniah, S. L., & Mahrin, M. N. (2016). A review on factors influencing implementation of secure software development practices. *International Journal of Computer and Systems Engineering*, 10(8), 3032–3039.
- Kanniah, S. L., & Mahrin, M. N. (2018). Secure software development practice adoption model: A delphi study. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 10(2-8), 71–75.
- 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 and Its Applications*, 7(3).
- Kongsli, V. (2006). Towards agile security in web applications. In *Companion to the 21st ACM SIGPLAN symposium on object-oriented programming systems, languages, and applications* (pp. 805–808). ACM.
- Leffingwell, D. (2010). *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional.
- Maxwell, J. A. (2013). *Qualitative research design: An interactive approach* (Vol. 41). Sage publications.
- McGraw, G. (2004, March). Software security. *Security & Privacy*, 2(2), 80–83. doi:10.1109/MSECP.2004.1281254
- McGraw, G. (2006). *Software Security: Building Security In*. Addison-Wesley.
- McGraw, G., Miguez, S., & West, J. (2018). *BSIMM 9*. Synopsys, Inc.
- Microsoft. (2009, June 30). Security development lifecycle for agile development, version 1.0.
- Microsoft. (n.d.). Microsoft security development lifecycle (No. Accessed 2019.08.07). Retrieved from <https://www.microsoft.com/en-us/SDL>
- Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook* (2nd ed.). Sage.
- Muneer, S. U., Nadeem, M., & Kasi, B. (2019). Comparison of modern techniques for analyzing NFRs in Agile: A systematic literature review. *Journal of Software Engineering Practice*, 3(3), 1–12.
- Nicolaysen, T., Sassoon, R., Line, M. B., & Jaatun, M. G. (2010). Agile software development: The straight and narrow path to secure software? *International Journal of Secure Software Engineering*, 1(3), 71–85. doi:10.4018/jsse.2010070105
- Oueslati, H., Rahman, M. M., & ben Othmane, L. (2015). Literature review of the challenges of developing secure software using the agile approach. In *Proceedings of the 10th international conference on availability, reliability and security (ARES)* (pp. 540–547).
- OWASP. (n.d.). Software assurance maturity model - a guide to building security into software development. version 1.5 (Tech. Rep.). *Open Web Application Security Project*.
- Peeters, J. (2005). Agile security requirements engineering. In *Proceedings of the Symposium on requirements engineering for information security*. Academic Press.
- Pohl, C., & Hof, H.-J. (2015). Secure scrum: Development of secure software with scrum.
- Poller, A., Kocksch, L., Türpe, S., Epp, F. A., & Kinder-Kurlanda, K. (2017). Can security become a routine?: a study of organizational change in an agile software development group. In *Proceedings of the 2017 ACM conference on computer supported cooperative work and social computing* (pp. 2489–2503). Academic Press. doi:10.1145/2998181.2998191
- Rajba, P. (2018, August). Challenges and mitigation approaches for getting secured applications in an enterprise company. In *Proceedings of the 13th International Conference on Availability, Reliability and Security* (pp. 1-6). Academic Press. doi:10.1145/3230833.3233276
- Renatus, S., Teichmann, C., & Eichler, J. (2015). Method selection and tailoring for agile threat assessment and mitigation. In *Proceedings of the 10th international conference on availability, reliability and security (ARES)* (pp. 548–555). Academic Press. doi:10.1109/ARES.2015.96

Rindell, K., Hyrynsalmi, S., & Leppänen, V. (2016, August). Case study of security development in an agile environment: building identity management for a government agency. In *Proceedings of the 2016 11th International Conference on Availability, Reliability and Security (ARES)* (pp. 556-563). IEEE. doi:10.1109/ARES.2016.45

Robson, C. (2011). *Real World Research* (3rd ed.). John Wiley & Sons.

Sachdeva, V., & Chung, L. (2017, January). Handling non-functional requirements for big data and IOT projects in Scrum. In *Proceedings of the 2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence* (pp. 216-221). IEEE. doi:10.1109/CONFLUENCE.2017.7943152

Saldanha, L. R., & Zorzo, A. (2019). Security requirements in agile software development: a systematic mapping study. Pontifical Catholic University of Rio Grande Do Sul, 2019, 32p.

Savola, R. M., Frühwirth, C., & Pietikäinen, A. (2012). Risk-driven security metrics in agile software development-an industrial pilot study. *J. UCS*, 18(12), 1679–1702.

Terpstra, E., Daneva, M., & Wang, C. (2017). Agile practitioners' understanding of security requirements: Insights from a grounded theory analysis. In *Proceedings of the 2017 IEEE 25th international requirements engineering conference workshops (REW)* (pp. 439–442). IEEE Press.

Tøndel, I. A., Jaatun, M. G., Cruzes, D. S., & Moe, N. B. (2017). Risk centric activities in secure software development in public organisations. *International Journal of Secure Software Engineering*, 8(4), 1–30. doi:10.4018/IJSSE.2017100101

van der Heijden, A., Broasca, C., & Serebrenik, A. (2018, October). An empirical perspective on security challenges in large-scale agile software development. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 1-4). Academic Press. doi:10.1145/3239235.3267426

Villamizar, H., Kalinowski, M., Viana, M., & Fernández, D. M. (2018, August). A systematic mapping study on security in agile requirements engineering. In *Proceedings of the 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 454-461). IEEE. doi:10.1109/SEAA.2018.00080

Williams, L., Gegick, M., & Meneely, A. (2009). Protection poker: Structuring software security risk assessment and knowledge transfer. In *Proceedings of the International symposium on engineering secure software and systems* (pp. 122–134). Academic Press.

Williams, L., Meneely, A., & Shipley, G. (2010). Protection poker: The new software security game. *IEEE Security and Privacy*, 8(3), 14–20. doi:10.1109/MSP.2010.58

*Inger Anne Tøndel is a PhD candidate at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway, and a research scientist at SINTEF Digital in Trondheim, Norway. She received the M.Sc. degree in Telematics from NTNU in 2004. Her research interests include software security, security requirements, information security risk management, cyber-insurance and smart grid cyber security.*

*Martin Gilje Jaatun is a Senior Scientist at SINTEF Digital in Trondheim, Norway. He graduated from the Norwegian Institute of Technology (NTH) in 1992, and received the Dr. Philos degree in critical information infrastructure security from the University of Stavanger in 2015. He is an adjunct professor at the University of Stavanger, and was Editor-in-Chief of the International Journal of Secure Software Engineering (IJSSE). Previous positions include scientist at the Norwegian Defence Research Establishment (FFI), and Senior Lecturer in information security at the Bodø Graduate School of Business. His research interests include software security, security in cloud computing, and security of critical information infrastructures. He is vice chairman of the Cloud Computing Association (cloudcom.org), vice chair of the IEEE Technical Committee on Cloud Computing (TCCLD), an IEEE Cybersecurity Ambassador, and a Senior Member of the IEEE.*

**Paper E: ‘The Security Intention Meeting Series as a way to increase visibility of software security decisions in agile development projects’**

The published material [31] is included here in accordance with ACM author rights.

E

# The Security Intention Meeting Series as a way to increase visibility of software security decisions in agile development projects

Inger Anne Tøndel

Department of Computer Science, Norwegian University  
of Science and Technology (NTNU)  
Trondheim, Norway  
inger.anne.tondel@ntnu.no

Daniela Soares Cruzes

Martin Gilje Jaatun  
Kalle Rindell  
SINTEF Digital  
Trondheim, Norway

## ABSTRACT

To achieve a level of security that is just right, software development projects need to strike a balance between security and cost. This necessitates making such decisions as to what security activities to perform in development and which security requirements should be given priority. Current evidence indicates that in many agile development projects, software security is dealt with in a more or less “accidental” way based on individuals’ security awareness and interest. This approach is unlikely to lead to an optimal security level for the product. This paper suggests *Security Intention Recap Meetings* as a recurring organisational tool for evaluating current practices regarding the security intentions of a software project, and to make decisions on how to move forward. These meetings involve key decision makers in the project, such as the product owner and the project manager, with the purpose of making security decisions visible and deliberate and to monitor their results.

## CCS CONCEPTS

• Security and privacy → Software security engineering; • Software and its engineering → Agile software development; Requirements analysis;

### ACM Reference Format:

Inger Anne Tøndel, Daniela Soares Cruzes, Martin Gilje Jaatun, and Kalle Rindell. 2019. The Security Intention Meeting Series as a way to increase visibility of software security decisions in agile development projects. In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES 2019) (ARES '19)*, August 26–29, 2019, Canterbury, United Kingdom. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3339252.3340337>

## 1 INTRODUCTION

In agile development projects, requirement management is dynamic. As a rule, a development project will not be able to deliver a perfect product within the cost and time constraints [15]. This makes requirements negotiation a key activity. In such an ecosphere, the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

ARES '19, August 26–29, 2019, Canterbury, United Kingdom

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-7164-3/19/08...\$15.00

<https://doi.org/10.1145/3339252.3340337>

security work needs to compete for its share of effort and money. Achieving cost-effective security, however, is not an easy task: Understanding and assessing the security needs of the software being under development is challenging in and of itself, further complicated by the complex and constantly changing threat landscape. Without a clear approach to identifying security needs and making decisions on how to address them, software projects are unlikely to end up with cost-effective security.

Studies have found evidence that software security is often dealt with in an “accidental” way in agile projects [23]. It has even been pointed out that “[a]gile techniques are vulnerable for forgetting things like security.” [22]. Security and quality aspects have a tendency to be sacrificed in favour of implementing more functionality [2, 22, 23], and the decisions involved are commonly made without involving security expertise [23]. The responsibility for software security is often unclear [22, 23] in projects and organisations. Thus, security is not a strategic decision, but rather left up to the individuals involved and their security posture. In particular, the Product Owner has been identified in studies as a common hindrance for sufficiently prioritizing security and quality [2, 22].

Security needs to be considered from the start and throughout a software project, and be visible as an important concern. This is acknowledged in various software security approaches [9, 17] and is reasserted by recent regulation regarding the handling of personal data [7]. Security decisions include decisions on which security activities and practices to perform and what security functionality to implement, but also include other decisions (e.g. design choices) that may have an impact on the security of the produced software. Security decisions are not only made in the beginning of the project, or only at some clearly identified gates, but happen throughout development in big and small ways, sometimes without security being explicitly taken into account.

This paper suggests an approach to bring security priorities and decisions forward in an agile development projects: the *Security Intention Recap Meeting*. This approach is based on ongoing interactions with several development companies [6] and on studies of other security techniques placed in an agile setting, in particular threat modeling [21] [10] and the Protection Poker risk estimation game [26][24]. The security intention recap meeting approach is made for the context of agile software development and project management. These meetings help addressing software security in a systematic way by involving the key decision makers of the project in regular assessments of the current state of the security work, and by comparing how the work is in line with the security



intentions for the project. The need to find a balance between security and cost is fully appreciated, and it is advocated that such a balance is unlikely to be achieved without intentional discussions about the right balance for this particular project.

This paper is organised as follows: Section 2 gives an overview of challenges identified in literature on prioritization of software security in agile software development projects. Section 3 presents the concept of the security intention recap meetings and how to organise them into a meeting series. Section 4 discusses the security intention meeting series in relation to other software security activities that are commonly recommended and can have similar goals; that is, threat modeling and risk assessments. Additionally the section identifies and discusses envisioned challenges to applying the approach in practice, explores ways to meet the challenges, and describes plans for future research. Section 5 concludes the paper.

## 2 CHALLENGES IN PRIORITIZING SECURITY IN AGILE SOFTWARE DEVELOPMENT

In this section, an overview of related studies is given, providing an empirical and theoretical basis for understanding the challenges of getting security prioritised in agile software development. The studies are generally in agreement that security is often neglected or underprioritized [2, 18, 22, 23] and point out many factors and challenges impacting how the security priorities are set in the agile development. The following factors are recurrent in various forms:

- The *individuals* have a key role and their attitude, knowledge, and priorities shape the priorities security is given in the development project. The product owner in particular influence priorities [2, 22], but there are also other important roles (security experts, developers and management at various levels) [13, 14] and there can be tensions between different groups [22, 23].
- The *ownership and responsibilities* for software security are currently unclear [22, 23]. This appears to have a negative impact on the priority given to security [23].
- The *business case* for security is unclear, and the security work is considered a fight not worth fighting [22]. The push for functionality is strong, and this results in less focus on security [2, 22, 23].

In a review of 44 primary studies, Kanniah and Mahrin [13] identified commonly cited factors impacting the successful implementation of secure software development practices. The broad set of factors identified include the institutional context, the people involved and their actions, the project content and the system development process. In a follow-up study with eight experts, the following factors were identified by Kanniah and Mahrin as the ten most important ones: 1) security experts, 2) security documentation, 3) project management, 4) developers, 5) project team, 6) security audit team, 7) team collaboration, 8) development time, 9) policy enforcement and 10) top management [14].

In a literature review of quality requirements work in agile development, Alsaqaaf et al. [2] identified the product owner as a hindrance for quality requirements being properly addressed. The product owners commonly have a “heavy workload” and “insufficient availability”, in addition to a “lack of knowledge” of quality

aspects [2]. Other challenges include inadequate or lacking techniques, and challenges that functionality is prioritized while some other types of requirements are ignored or insufficiently analysed.

In their systematic literature review, Oueslati et al. identified 14 challenges of developing secure software within the agile approach [18]. The challenges were categorised the following way:

- “*Software development life-cycle challenges*”: security activities not included; hard to integrate security in every iteration due to short iteration times.
- “*Incremental development challenges*”: dealing with changes.
- “*Security assurance challenges*”: documentation; testing; unstable development process.
- “*Awareness and collaboration challenges*”: security requirements neglected; lack of experience and security awareness; separate the developer and reviewer roles.
- “*Security management challenges*”: giving priority to security.

Tøndel et al. [23] studied software security practices among 23 public organisations, using interviews as the main instrument of data collection. This study aimed to identify risk-centric software security practices in organisations and projects in the public domain. It involved people in development and information security positions in organisations mainly using some type of agile software development practices. The findings show that software security work in these organisations is not generally based on security risk, but rather triggered by the requirements for legal and regulatory compliance, or more or less “accidental” detection of security mistakes in development. Barriers against security include unclear responsibilities for software security, architects without an interest in security, lack of security knowledge both on the developer and procurer side, and security being considered a “technical issue”. In the organisations studied, it was found that “[n]o one fights for software security”, that risk treatment decisions were often “[a]rbitrary, late and error driven” and that “[t]ime pressure results in security requirements being postponed (or even dropped)” [23].

Terpstra et al. [22] studied practitioners’ postings on social media (LinkedIn) to discover how agile practitioners reason about security requirements, and how they cope with them. The analysis resulted in the identification of 21 concepts that indicate problems regarding security requirements in agile, and 15 coping strategies. Problems identified include the limited business value of security, the tendency that security gets lost in the process, and the lack of awareness and knowledge. Their analysis resulted in a descriptive conceptual framework that included the following categories:

- “[O]wnership of security requirements”: represents the finding that no role assumes, or is given, full responsibility for security requirements in development projects.
- “[D]efinition of Done (DoD)”: represents the opinion of some professionals that the DoD should represent requirements on the need to implement security measures.
- “[B]usiness case”: represents the findings pointing to security not being part of the project’s business case.
- “[A]ttitude towards security requirements”: represents the findings that in some cases “team members ‘do not care’ about security requirements just because there is no incentive to do so (...). Or, because no one really understands completely what these requirements are”.

- “[O]rganizational setup”: represents the findings that show that the organisational culture can both help and hurt the security requirements work. In particular approaches to educating developers on software security could have an impact.
- “[P]erceptions of priority”: represents issues related to prioritisation of security requirements at inter-iteration time. Business representatives often drive priorities, pushing for functionality, but their priorities can differ from developers.

### 3 THE SECURITY INTENTION MEETING SERIES

A security intention (SI) recap meeting is a meeting primarily for decision makers, and is intended to be part of a series where the meetings build on each other. Both these aspects of the meeting are necessary to reach the meeting goal of making software security decisions more visible, systematic and deliberate. In the following we explain how the SI recap meetings are organised into a series and integrated into development before we move on to explaining the different parts of the SI recap meeting in more detail

#### 3.1 Integration into development

How often SI recap meetings should be held would depend on the product and may vary throughout the project life cycle. Note, however, that the SI recap meetings are meant to be relatively short meetings (ideally maximum one hour), and we advocate to rather have short meetings more often than longer meetings more seldom. Figure 1 gives one example of possible timing in relation to the software development activities of the project.

We envision one initial SI meeting in the beginning of the project, one SI postmortem meeting in the end, and several SI recap meetings during the course of the project. In the initial SI meeting at the beginning of the project, the goal of the meeting is to clarify the overall goals of the software security work in this project, decide on which statements to use for self-evaluation during the project, and decide on initial security activities needed in the initiation of the project. The goal of the SI postmortem meeting is to evaluate the software security approach in this project, related to the goal, and identify learning points for future development projects. The SI postmortem meeting could utilize any postmortem technique [4] and could be part of a larger postmortem meeting for the project, covering more issues than software security.

The SI meeting series, in addition to making software security decisions more visible and deliberate, offers a possibility to document important assumptions, priorities and decisions regarding software security throughout the project. Thus notes should be taken from the meetings and stored as part of the project documentation. Action points from the meeting should find their way into any tools used for issue tracking. The self-evaluation results additionally offer a way to track progress throughout the project and learn more about what types of actions create the effects sought after in order to meet software security goals.

#### 3.2 The SI recap meeting

The SI recap meeting consists of two main parts: 1) an honest evaluation of the current state of software security in the development of the product, and 2) deciding on action points on how to move

<b>INTENTION:</b> In this project security is important because [...] We want to make sure we deliver quality to our customer, and this includes delivering the right amount of security for their needs.			
Meeting date and person responsible	People present		
<b>STATUS ASSESSMENT:</b> We want to know the current state of our approach to software security, so that we can make good decisions on how we will move forward from here Score: great, good, somewhat, lacking			
Statement	Score	Successes	Opportunities to improve
The teams have the skills to understand and address security in the software			
We know what are the most relevant attackers and attack goals for the software we develop			
...			
<b>WAY FORWARD:</b> We will regularly improve our competence and ways of working, to ensure we work in line with our intentions Note: let these goals and steps be based on the current status, but also make sure to revisit the goals and steps decided on in the previous security intention recap and consider what to keep, what to drop, what to change			
Improvement goal	Concrete step	Responsible	Plan for progress follow up
<b>Schedule and plan for next security intention recap</b>			
Date and person responsible	Who should join	Any improvements to the security intention recap meeting	

Figure 2: Template for the security intention recap meeting

from here. The meeting shall not be longer than one hour. Figure 2 gives an example template for an SI recap meeting. In the following we explain the key parts of the meeting.

3.2.1 *Owner and Participants.* As the SI recap meeting is a meeting that is about making conscious decisions on software security, it is of paramount importance that roles involved in making decisions related to the particular product under development participates in the meeting. Roles such as product owner and project manager should be part of the meeting. Additionally, roles with security responsibility, or responsibility for legal compliance (if relevant) should be part of the meeting. In addition to these roles, it is necessary to have meeting participants that are in touch with what is happening in the actual development. Thus, it may be decided to include one or more developers or testers in the meeting, in particular people with a security champion or a team leader role.

One person needs to be responsible for the SI recap meeting, and for keeping the SI meeting series alive. This person needs to be motivated about software security. Ideally this person should be part of the development project, so that the meeting is not seen as initiated from outside the project.

3.2.2 *Status assessment.* The heart of the SI recap meeting is an honest evaluation of the current state of software security in the development of this particular product. Doing such a self-evaluation



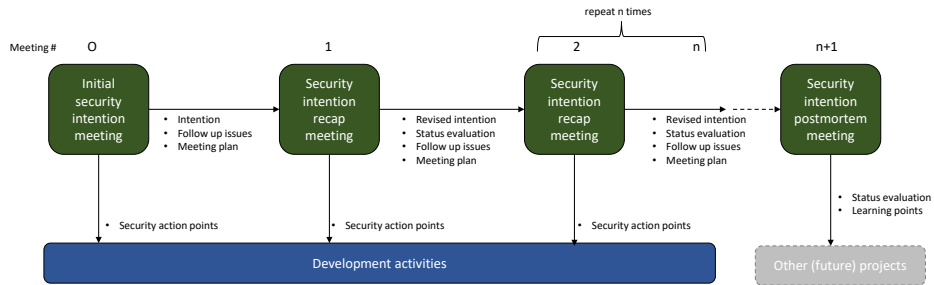


Figure 1: Illustration of the relationship between the security intention meetings and their output

serves two purposes: 1) to remind the participants of the security goals of the project and what level of software security that is aimed for, and 2) to identify areas where there is a need to adjust practices to be more in line with the software security intentions for the product. Current practice is evaluated related to a set of previously selected statements (see subsection 3.2.3) that concretise what level of software security is aimed for in this product. For each statement the participants discuss successes and opportunities to improve (inspired by a tool for assessing onboarding [3]). Additionally, we recommend that the meeting participants evaluate each of the statements according to the following scale, to be clear about where the practices are acceptable and where improvement is needed:

- *Great*: We are doing great, and do not need to prioritize further improvement in this area
- *Good*: We know we could be better, but are fairly satisfied with current practice
- *Somewhat*: We are doing some things, but really should improve this part
- *Lacking*: We are doing close to nothing and are far from realizing this goal

Discussing and documenting both an evaluation of current practice and the basis for this evaluation (successes; opportunities to improve) is the foundation for making decisions on the way forward.

In the self-evaluation part of the meeting it is essential that all participants have their say so that the self-evaluation ends up being as true as possible related to the current state. One wants to avoid one or two meeting participants dominating the evaluation, leaving out other perspectives. It is possible to use a voting mechanism similar to that used in Planning Poker [8] or Protection Poker [25, 26] to ensure all participants make an individual assessment of each statement, and that each individual assessment is made visible in the meeting. In any case, the moderator of the SI recap meeting is essential in creating a safe atmosphere for discussion and making sure all relevant voices are heard in the meeting.

**3.2.3 Process for selecting statements for self-evaluation.** Each project should select a manageable set of statements to assess for the project. Selecting statements for self-evaluation is a way of making priorities for the project, as these statements will be used to

guide attention and decisions in the project. Selecting some statements implies not selecting others. As priorities need to be made related to the particular project, we do not provide a finished list to choose from, but rather a process for selecting statements. The self-evaluation statements should be decided on in the beginning of the project, but can be revised as the project moves along if underlying assumptions or overall priorities change related to software security, or if one has reached the goals on one statement and wants to put the focus elsewhere. The template in Figure 2 shows example content for the self-evaluation statements.

Statements can be identified top-down, based on priorities on an organisational level, or bottom-up, based on the individual project, or a combination. We suggest that the first step in deciding on a set of self-evaluation statements is to answer the following questions:

- Does the organisation have any strategies that sets out the goals or ambitions regarding software security? Examples of such documents would be software security manifestos or KPIs.
- Does the organisation know its strong and weak spots when it comes to software security, and have identified areas of improvement? Examples would be results from a BSIMM or OpenSAMM evaluation of practices.
- Does the product being developed have any specific characteristics that can influence software security and make software security different than in most other products we develop? Examples could be customer expectations, legal requirements, technology, and exposure of the software.
- Are there any aspects of the team(s) involved that impact our ability to do software security well? Examples could be security competence and awareness, and team culture.

By answering the above questions one would identify the main sources for the self-evaluation statements. The next step would be to identify possible statements from these main sources. Then the final step would be to choose a manageable set of self-evaluation statements for the project. We suggest to start with 5 to 7 statements. If assuming that five minutes would be enough for an evaluation of each statement, that would imply from 25 to 35 minutes of the meeting spent on status assessment.

Projects need to balance the need for having self-evaluation statements that are something to aim for and give a motivation to improve, and the need for realistic statements. We recommend that the statements selected represent the real ambitions of the project, meaning that if a statement is met then the project is at the right level of security in that area.

To give an idea of how self-evaluation statements can be identified in practice, Table 1 shows potential self-evaluation statements that have been made based on the DevSecOps manifesto inspired from the Build-Security-In Manifesto and the Secure Development Lifecycle initiative principles at Comcast [16], BSIMM scores from an evaluation of software security maturity in public organisations [11] and for applications that handle personal health information that is subject to legal requirements [12].

**3.2.4 Way forward.** Based on the status assessment, the participants in the SI recap meeting should decide on concrete action points that would move the state of software security more in line with the goals for the product. Note that this may mean to start or improve some software security initiatives (e.g. do a threat modeling session, do a training session on a specific software security topic, do a risk assessment, increase security testing efforts, etc.), but it can also mean to stop or reduce efforts in one or more existing software security activities. The action points decided on need to be concrete and have a deadline and someone responsible in order to increase likelihood that the action points will be followed up in the day-to-day development activities. To increase the commitment to the action points, we would suggest that the SI recap meeting participants, as part of the “way forward” part of the meeting, revisit decisions from the previous meetings to see if the previous action points have been followed up, and if not, discuss how to increase the likelihood that the action points decided upon in the current meeting will have more of an impact.

One important part of deciding on the way forward related to software security is to decide upon when the next SI recap meeting should be held for this product, and who should participate. The reason we suggest that this is decided upon in this meeting, and put into the calendars of the participants, is to reduce the likelihood that the commitment to having these meetings is forgotten.

## 4 DISCUSSION

This section explains how the SI recap meeting is complementary to other software security techniques such as risk assessment and threat modeling. It moves on to looking at some known challenges related to adoption of threat modeling and a risk estimation technique called Protection Poker. These already identified challenges are then used to describe likely challenges to the SI recap meetings so that these challenges can be proactively addressed. Finally, we describe future research endeavors to evaluate and improve the SI recap meeting approach.

### 4.1 Relation to other software security techniques

Threat modeling [21] and risk assessment [23] activities can be used to make decisions and priorities on how to move forward based on an assessment of the current status. Compared to an SI recap meeting, the status assessments made in these types of

activities are normally on a much lower level of abstraction and with an emphasis on the system and what can go wrong. Thus these activities usually take longer than what is envisioned for an SI recap meeting. Often these activities are performed by participants with technical competence and leave out decision makers. The SI recap meetings are not an alternative to risk assessment or threat modeling, but rather a place where the decision to perform or not perform risk assessment or threat modeling activities could be made.

Protection Poker [25, 26] is a security risk estimation game that is particularly suited for agile teams. It offers a practical way of doing risk assessment in an iterative fashion, and looks at the assets and the ease of attack that comes with implementation of features. Compared to the SI recap meetings, Protection Poker’s goal is less geared towards decision making. The participants are different, with Protection Poker involving the entire team. The time it takes to play Protection Poker can vary from team to team, depending on the discussions and their familiarity with the game. However, as Protection Poker is intended to be played for every iteration with the full team, the total time it takes would likely be much longer than an SI recap meeting.

### 4.2 Potential Challenges

The SI recap meeting has not yet been tried out in practice in development companies. The suggested approach is a response to reported challenges in literature on having security being given the “right” priority in agile software development projects, as well as our own experiences with ongoing interactions with software companies on software security [6]. Though the SI recap meetings are different than techniques such as threat modeling and Protection Poker, we believe it would face some of the same challenges to adoption. Thus we have looked to a study of adoption of Protection Poker [24] and a study on applying Microsoft Threat Modeling to agile projects [5] to identify what we believe are likely challenges to adoption of the SI recap meeting approach. In the following we explain these envisioned challenges and provide suggestions for how to address them. Table 2 gives an overview of how challenges identified for Protection Poker and threat modeling relate to the envisioned challenges to the SI recap meetings.

**4.2.1 Perceiving improved software security as a consequence of the SI recap meeting.** It is a likely challenge that the SI recap meeting is viewed as “yet another meeting”. For both Protection Poker and threat modeling it was challenging to see clearly how the technique led to improved security of the software, and not only discussions about security. The SI recap meeting, is likely to face the challenge of having a visible and traceable direct impact on the delivered security of the code.

To address this challenge, the SI recap meeting needs to ensure that the meeting leads to actionable decisions that are followed up in development. Having participants with the authority to make decisions and have them implemented is thus of key importance. At the same time, it is important that the development team(s) have confidence that the decisions reached in these meetings are good ones. Thus the competence of the participants is important, both related to security and the overall understanding of the product, as

Source	Issue	Potential self-evaluation statements
DevSecOps Manifesto [16]	<p><i>"Build security in more than bolt it on"; "Implement features securely more than security features"</i></p> <p><i>"Rely on empowered engineering teams more than security specialists"; "Build on culture change more than policy enforcement"</i></p> <p><i>"Use tools as feedback for learning more than end-of-phase stage gates"</i></p>	<ul style="list-style-type: none"> <li>• We consider security in the design of all functionality, not only for security features.</li> <li>• The teams have the skills to understand and address security in the software.</li> <li>• The teams feel responsible for how the software behaves in production, including security implications.</li> <li>• We use security testing tools early to give feedback to developers and improve their skills in writing vulnerability-free software.</li> </ul>
BSIMM scores [11]	<p>Attack Models is the area with lowest maturity</p> <p>Strategy and Metrics is the area with the second lowest maturity</p>	<ul style="list-style-type: none"> <li>• We know what are the most relevant attackers and attack goals for the software we develop.</li> <li>• We have a clear processes for software security, and this process is known and followed by the development team.</li> </ul>
Product characteristics	Health information	<ul style="list-style-type: none"> <li>• We meet legal requirements for protection of health related data</li> </ul>

Table 1: Example self-evaluation statements and their sources

Envisioned SI recap meeting challenge	Related Protection Poker (PP) challenges [24]	Related Threat Modeling challenges [5]
Perceiving improved software security as a consequence of the SI recap meeting	<ul style="list-style-type: none"> <li>• PP did not improve security of the software</li> <li>• Ensuring confidence in the results</li> <li>• Important aspects from the discussion is lost</li> <li>• The output from playing PP is not concrete in terms of what to do next</li> </ul>	<ul style="list-style-type: none"> <li>• Documentation of the assets after the meeting was not done</li> <li>• Many discussions on threats and mitigation strategies get lost</li> <li>• The approach does not make a link to the actual code</li> <li>• It is hard to know when enough analysis has been done</li> <li>• The output of the sessions are a list of concerns/threats that are not concrete</li> <li>• Follow up of the threats is challenging</li> </ul>
Running and facilitating the meeting	<ul style="list-style-type: none"> <li>• It is difficult to reach consensus, something that results in a lot of time spent and sometimes results in tension in the team</li> <li>• Some team members may end up with too much influence</li> </ul>	<ul style="list-style-type: none"> <li>• The meeting needs to be structured, but it is not always clear on how to run the meeting</li> <li>• It is hard to know which other people should be included in the meetings besides the "core" development team</li> <li>• There are challenges with running meetings in distributed settings</li> <li>• The meetings are not effective</li> </ul>
Selecting self-evaluation statements	<ul style="list-style-type: none"> <li>• Starting to use PP is time consuming due to calibration and the need to identify and play about assets</li> <li>• Selecting granularity of assets and assigning value to assets can be challenging</li> </ul>	<ul style="list-style-type: none"> <li>• It is challenging to motivate the teams to draw the diagrams</li> <li>• It was hard to decide on the right level of abstraction to the DFDs</li> <li>• It takes long time to draw the diagrams</li> </ul>
Establishing the SI recap meeting as a regular event	<ul style="list-style-type: none"> <li>• Teams did not end up using PP in a regular fashion</li> <li>• Planning meetings are already full</li> <li>• PP takes too much time</li> </ul>	<ul style="list-style-type: none"> <li>• There is a need for a security expert to run the meeting; not every team has this profession available</li> <li>• It is not easy to have everyone participating</li> </ul>

Table 2: Selected challenges from study of Protection Poker[24] and Microsoft Threat Modeling [5]

well as how the decisions made and their rationale is communicated outside of the SI recap meeting.

**4.2.2 Running and facilitating the meeting.** Having effective security meetings where everybody's opinion is heard and valued, while at the same time aiming to reach some kind of consensus, is challenging. As in the Protection Poker meetings [24], at the SI recap meetings there will likely be some participants with more authority than others, and there is a risk that these may end up influencing other participants to the extent that important perspectives are lost. This risk comes in addition to common challenges on how to run meetings, how to know who should participate and how to deal with distributed settings, as was found for threat modeling [5].

Addressing these challenges fully is difficult, as it involves balancing somewhat conflicting goals (efficiency vs. including many perspectives). However, having a skilled facilitator would be an important step in ensuring quality of the meetings themselves.

**4.2.3 Selecting self-evaluation statements.** Protection Poker and threat modeling approaches do not use self-evaluation statements as we propose for the SI recap meetings. However, they need other kinds of preparations (for example, calibration and possibly asset identification and evaluation for Protection Poker, Data Flow Diagrams (DFDs) for threat modeling). These preparations require an upfront investment in time and effort. Experiences from Protection Poker and threat modeling show that it can be challenging to motivate participants for these preparatory activities, and that they can be perceived as time consuming. The tasks can additionally be challenging when it comes to "doing them right", e.g. at the right level of abstraction that makes them useful in the upcoming activities.

As has already been pointed out, identifying self-evaluation statements is a challenging task, and one that is important as it guides future priorities. It is likely that this preparatory task will require time and effort from key people if projects are to arrive at an optimal set of self-evaluation statements. It is hard to foresee how this will play out in practice before we start experimenting with it in real companies and software development projects. We expect that the process for selecting self-evaluation statements that is laid out in subsection 3.2.3 will be improved substantially based on future experiences with this part of the approach.

**4.2.4 Establishing the SI recap meeting as a regular event.** The SI recap meetings are intended to be regular events that will serve as reminders of security commitments and increase visibility of security decisions throughout the development of an application. Making the SI recap meetings into regular events is important to increase visibility and awareness of all the ways security decisions are made throughout the project, and influence these in a more deliberate way. However, establishing a new regular practice is challenging. It takes commitment from the project team over a long period of time. In the studies of Protection Poker and threat modeling, time issues and having people participate was considered challenging. Another study of Protection Poker found that although the technique was found to have important benefits, the team still stopped using Protection Poker some time after the study for unknown reasons [26].

As already pointed out in section 3, it is important that one person is responsible for the SI recap meeting and for keeping the SI meeting series alive, "championing it". Finding such a person, that is both interested enough in software security and has the necessary influence on the development project and ability to motivate others, can however be challenging. Not every project may have such a person available.

The SI meeting series we propose is by design lightweight and can be easily adjusted to the needs of the project and the organisation. Adjustments can be made regarding meeting schedule, participants and self-evaluation statements, something that may ease adoption of the technique. Additionally, it is possible to argue that the meeting has the potential to save effort in the longer run, as making more deliberate security decisions and priorities may lead to reduced costs later on (e.g. through not spending time on more security activities than necessary, and by reducing the need for expensive changes that stem from big "surprises" related to security implications of features or design choices). This however does not take away the need for someone that is willing and able to push for the meeting in a way that actually leads to longer-term adoption.

### 4.3 Further Work

Though the SI recap meeting approach builds on challenges and needs that have been identified in previous research and in our own continuing collaboration with software security companies [6], the approach itself has not yet been tried out and validated empirically. In the future, we plan to foster the adoption of this approach in some of the companies we collaborate with and then collect experiences and improve the approach. In particular we are interested in investigating the following research questions:

- **Adoption:** How is the approach received in the companies? What makes them interested in adopting the approach, and how can one support long-term adoption?
- **Effects:** How does the SI recap meeting influence the development, and what can be done to increase the positive effects of using this approach while minimising the cost?
- **Support:** How can agile projects be supported in using the SI recap meeting approach? Where is support most needed and what are the recommendations that are most important to give to projects wanting to adopt the approach when it comes to frequency, length of meeting and participants?
- **Self-evaluation statements:** How to select statements in a way that is motivating? How can projects be supported with recommendations of what is important statements for their kind of projects?

Additionally, we would welcome more research that can contribute to understanding what can be done to support longer term adoption of software security activities and that can help understand what are the most important goals and practices to strive for in an agile project when it comes to software security. Better understanding of these aspects can feed the content of the SI recap meeting, but also software security priorities in a more general sense.



## 5 CONCLUSION

This paper proposes the security intention meeting series as a way to make software security decisions more visible, systematic and deliberate in agile development projects. By tackling current challenges that security is easily “forgotten” in agile development [22] and sacrificed for functionality [2, 22, 23] and that security priorities are highly dependent on the varying interests of the individuals involved [2, 22], projects are more likely to move towards cost-effective software security. In order to achieve this, the SI recap meetings needs to be adopted by software projects and organisations and integrated into their way of working.

## ACKNOWLEDGMENT

This work was supported by the *Science of Security in Agile Software Development* project (SoS-Agile), funded by the Research Council of Norway (grant number 247678).

## REFERENCES

- [1] Icek Ajzen. 1991. The theory of planned behavior. *Organizational Behavior and Human Decision Processes* 50, 2 (1991), 179 – 211. [https://doi.org/10.1016/0749-5978\(91\)90020-T](https://doi.org/10.1016/0749-5978(91)90020-T) Theories of Cognitive Self-Regulation.
- [2] Wasim Alsaqaf, Maya Daneva, and Roel Wieringa. 2017. Quality requirements in large-scale distributed agile projects—a systematic literature review. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 219–234.
- [3] Talya N Bauer. 2010. Onboarding new employees: Maximizing success. *SHRM Foundation's Effective Practice Guideline Series* 7 (2010).
- [4] Paulo Caroli and Taina Caetano. 2015. *Fun Retrospectives - Activities and ideas for making agile retrospectives more engaging*. Leappub, Layton.
- [5] Daniela Soares Cruzes, Martin Gilje Jaatun, Karin Bernsmed, and Inger Anne Tøndel. 2018. Challenges and Experiences with Applying Microsoft Threat Modeling in Agile Development Projects. In *2018 25th Australasian Software Engineering Conference (ASWEC)*. IEEE, 111–120.
- [6] Daniela S. Cruzes, Martin G. Jaatun, and Tosin D. Oyetoyan. 2018. Challenges and Approaches of Performing Canonical Action Research in Software Security: Research Paper. In *Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security (HotSoc'18)*. ACM, New York, NY, USA, Article 8, 11 pages. <https://doi.org/10.1145/3190619.3190634>
- [7] EU. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). L 119 (2016).
- [8] James Grenning. 2002. Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting* 3 (2002), 22–23.
- [9] Michael Howard and Steve Lipner. 2006. *The Security Development Lifecycle*. Microsoft Press.
- [10] Martin Gilje Jaatun, Karin Bernsmed, Daniela S. Cruzes, and Inger Anne Tøndel. 2019. Threat Modeling in Agile Software Development. In *Exploring Security in Software Architecture and Design*, Michael Felderer and Riccardo Scandariato (Eds.). IGI Global.
- [11] Martin Gilje Jaatun, Daniela S. Cruzes, Karin Bernsmed, Inger Anne Tøndel, and Lillian Rastad. 2015. Software Security Maturity in Public Organisations. In *Information Security*, Javier Lopez and Chris J. Mitchell (Eds.). Lecture Notes in Computer Science, Vol. 9290. Springer International Publishing, 120–138.
- [12] J. Jensen, I. A. Tøndel, M. G. Jaatun, P. H. Meland, and H. Andresen. 2009. Reusable Security Requirements for Healthcare Applications. In *2009 International Conference on Availability, Reliability and Security*. 380–385. <https://doi.org/10.1109/ARES.2009.107>
- [13] Sri Lakshmi Kanniah and Mohd Naz'ri Mahrin. 2016. A review on factors influencing implementation of secure software development practices. *World Academy of Science, Engineering and Technology, International Journal of Social, Behavioural, Educational, Economic, Business and Industrial Engineering* 10, 8 (2016), 2860–2867.
- [14] Sri Lakshmi Kanniah and Mohd Naz'ri Mahrin. 2018. Secure Software Development Practice Adoption Model: A Delphi Study. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 10, 2-8 (2018), 71–75.
- [15] Dean Leffingwell. 2010. *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional.
- [16] Larry Maccherone. 2017. The DevSecOps Manifesto. <https://medium.com/continuous-agile/the-devsecops-manifesto-94579e0eb716>. (2017). Accessed: 2019-04-30.
- [17] Gary McGraw. 2006. *Software Security: Building Security In*. Addison-Wesley.
- [18] Hela Oueslati, Mohammad Masudur Rahman, and Lotfi ben Othmane. 2015. Literature review of the challenges of developing secure software using the agile approach. In *10th International Conference on Availability, Reliability and Security (ARES)*. IEEE, 540–547.
- [19] James O Prochaska. 2008. Decision making in the transtheoretical model of behavior change. *Medical decision making* 28, 6 (2008), 845–849.
- [20] Ronald W Rogers and Steven Prentice-Dunn. 1997. Protection motivation theory. In *Handbook of health behavior research 1: Personal and social determinants*. Plenum Press, 113–132.
- [21] Adam Shostack. 2014. *Threat Modeling: Designing for Security*. Wiley.
- [22] Evenynke Terpstra, Maya Daneva, and Chong Wang. 2017. Agile Practitioners' Understanding of Security Requirements: Insights from a Grounded Theory Analysis. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*. IEEE, 439–442.
- [23] Inger Anne Tøndel, Martin Gilje Jaatun, Daniela Soares Cruzes, and Nils Brede Moe. 2017. Risk Centric Activities in Secure Software Development in Public Organisations. *International Journal of Secure Software Engineering (IJSSSE)* 8, 4 (2017), 1–30.
- [24] Inger Anne Tøndel, Laurie Williams, Daniela Soares Cruzes, and Martin Gilje Jaatun. 2019. Collaborative Security Risk Estimation in Agile Software Development. *Information and Computer Security* (2019).
- [25] Laurie Williams, Michael Gegick, and Andrew Meneely. 2009. Protection poker: Structuring software security risk assessment and knowledge transfer. In *International Symposium on Engineering Secure Software and Systems*. Springer, 122–134.
- [26] Laurie Williams, Andrew Meneely, and Grant Shipley. 2010. Protection poker: The new software security game. *IEEE Security and Privacy* 8, 3 (2010), 14–20.

## **Paper F: ‘Achieving “Good Enough” Software Security: The Role of Objectivity’**

The published material [32] is included here in accordance with ACM author rights.



F

# Achieving “good enough” software security: the role of objectivity

Inger Anne Tøndel  
inger.anne.tondel@ntnu.no  
Norwegian University of Science and  
Technology (NTNU)  
Trondheim, Norway

Daniela Soares Cruzes  
SINTEF Digital  
Trondheim, Norway  
daniela.s.cruzes@sintef.no

Martin Gilje Jaatun  
SINTEF Digital  
Trondheim, Norway  
martin.g.jaatun@sintef.no

## ABSTRACT

Today’s software development projects need to consider security as one of the qualities the software should possess. However, over-spending on security will imply that the software will become more expensive and often also delayed. This paper discusses the role of objectivity in assessing and researching the goal of good enough security. Different understandings of objectivity are introduced, and the paper explores how these can guide the way forward in improving judgements on what level of security is good enough. The paper recommends adopting and improving upon methods that include different perspectives, support the building of interactive expertise, and support confirmability by keeping documentation of the basis on which judgements were made.

## CCS CONCEPTS

• **Security and privacy** → **Software security engineering**; • **Software and its engineering** → **Agile software development**; **Risk management**; *Requirements analysis*.

## KEYWORDS

software security, objectivity, security level, good enough security, security priorities, agile software development

### ACM Reference Format:

Inger Anne Tøndel, Daniela Soares Cruzes, and Martin Gilje Jaatun. 2020. Achieving “good enough” software security: the role of objectivity. In *Evaluation and Assessment in Software Engineering (EASE 2020)*, April 15–17, 2020, Trondheim, Norway. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3383219.3383267>

## 1 INTRODUCTION

In today’s interconnected and digitized world, a large portion of the software that is developed needs to consider security. This is the case not only for software that is considered security critical (e.g., military systems), but also for the more “normal” type of software (e.g., web applications, mobile apps). The goal and focus of these software development approaches is to deliver value to

the customers, and security is commonly seen as a secondary goal; something that must be considered but usually would imply extra development time and costs. Viewed this way, security could have a negative impact on the output of functionalities, and over-spending on security may make the software less competitive or successful. However, not putting in the necessary security is not a good option, as this may cause severe problems later on.

Security experts in general would agree that perfect or total security is an illusion [13, 24], this is e.g. a foundation for a risk-management approach to security [13] - an approach taken in major security standards such as ISO/IEC 27001 [14]. Determining what is good enough is however hard [24]. Beznosov [5] suggests not defining what is “good enough”, but rather letting the customer define and adjust security needs as the project progresses, utilizing the customer involvement built into agile development approaches. Sandhu [24] suggests two design principles: “Designing with the application in mind”, and viewing security as being “about trade-offs, not absolutes” [24]. Hurlburt [13] points to the human factor as a reason why “systems security will never be better than good enough” [13]. He acknowledges the importance of investing in a robust upfront security design, but argues that this will not solve the problem completely. He points out that with today’s distributed attacks there is a need for more overarching approaches, not only considering whether an attacker might be discouraged from attacking one particular system. He suggests instead, “an objective, consensus-based rating system” [13] that companies can use to rank risks of different products and organizations, and claims that through the use of such a rating system one may establish a kind of working threshold that defines what is considered good enough when it comes to security.

Objectivity is a key characteristic for research in general, and even a part of research ethics [21]. Objectivity can be understood in different ways, but is often considered as a striving towards avoiding bias [9, 21] to ensure adequate “standing of our judgements and interpretations” [9]. Thus, objectivity is related to a confidence that multiple observers could come to similar judgements [22]. We agree with Hurlburt that there is a need to objectively assess what is good enough when it comes to security. We additionally agree with Sadhu that “good enough” is not something that can be defined outside of the context of a particular project, organization or product. In this emerging results paper we continue the discussion on objectivity in relation to security, exploring what it would mean to have an “objectively correct” level of security, and which security analysis approaches can support objective judgements about “good enough security”. We build on insights from our own empirical studies on software security performed over a period of several

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*EASE 2020, April 15–17, 2020, Trondheim, Norway*  
© 2020 Association for Computing Machinery.  
ACM ISBN 978-1-4503-7731-7/20/04...\$15.00  
<https://doi.org/10.1145/3383219.3383267>

years. Additionally, we introduce theory on objectivity and examine what this type of theory may bring to security practice and research.

The paper is organised as follows. Section 2 introduces the concept of objectivity in more detail, and different ways to understand objectivity. Section 3 explains the research approach taken in the empirical research that underlie the claims we make in this paper about software security practice in agile development projects. Then Section 4 exemplifies challenges in reaching objective evaluations of software security. In Section 5 we propose and discuss strategies that could help increasing objectivity both in software security analysis and in research on “good enough” software security. Section 6 concludes the paper.

## 2 AN INTRODUCTION TO DIFFERENT UNDERSTANDINGS OF OBJECTIVITY

In this paper we use a set of different philosophical understandings of objectivity, as described by Gaukroger [9], to structure our introduction to objectivity. Additionally, we bring in understandings and practical considerations on how to achieve objectivity from research methods literature.

Objectivity can be understood as “a judgement that is free of prejudice and bias” [9], or even as “a judgement which is free of all assumptions and values” [9]. These understandings are both describing a “particular state of mind” [9], and are both negative theories of objectivity, stating what should be removed in order to be objective. Aiming to be free of prejudice and bias can be a challenging endeavor, but Gaukroger argues that it is a sensible goal [9]: “Objectivity requires us to stand back from our perceptions, our beliefs and opinions, to reflect on them, and subject them to a particular kind of scrutiny and judgement. Above all, it requires a degree of indifference in judging that may conflict with our needs and desires.” [9]. Removing all assumptions and values is however not possible [9, 20]; nothing is a “view from nowhere” [9], all beliefs are socially situated [11].

Quantitative methods have long been associated with objectivity. There is a seeming neutrality that comes from having numbers - “The numbers speak for themselves!” [9]. Gaukroger argues that practices that could fall into the term ‘number crunching’ “are not necessarily subjected either to reasoned judgement or to the empirical evaluation of particular cases, but typically bypass any form of independent or objective reasoning at all” [9]. Aiming for a judgement free from prejudice and bias does not mean eliminating judgement. Instead of letting the numbers be “a substitute for decision making” they could be used as “an aid to decision making” [9], or even as illustrations allowing for a more persuasive argument [7].

Objectivity can be understood as consisting of “accurate representations” [9]. What constitutes an accurate representation is however subject to judgement, and can be considered differently depending on what one wants the representations for. Objectivity is not an absolute - you are not either objective or not objective - but rather there are degrees of objectivity. The understanding of objectivity as accurate representations points to objectivity as “something that can be learned and improved upon through practice” [9]. “Trained judgement” as well as “identification and elimination of arbitrary judgement” [9] becomes important traits of objectivity.

Objectivity as accurate representation however is costly and needs to be balanced against other concerns.

Objectivity can be understood related to the procedure used, viewing an objective procedure as “one that allows us to decide between conflicting views of theories” [9]. Objectivity is seen as a core aspect of science [9], and the methods used in science is expected to support objectivity. The scientific method and the progress towards better and better theories rely on theories being falsifiable and that scientists do serious attempts at refuting theories [20]. It has however been argued that the way objectivity is used in science does not necessarily fit other contexts, e.g. the needs when studying human behaviour [9] or when using other research paradigms [17] than those using conventional scientific methods. The types of methods used in e.g. sociology have some fundamentally ‘subjective’ traits where research approaches to eliminate subjectivity, such as double-blind testing, do not work [7]. This however does not mean that qualitative studies cannot strive for and demonstrate objectivity. In the following we introduce two examples of this: the suggestion to replace objectivity with confirmability in naturalistic studies [17], and the concept of strong objectivity from feminist studies [11].

Lincoln and Guba [17] argue that confirmability is a preferable concept to objectivity within the naturalistic paradigm, a primarily qualitative research paradigm where studies are performed in natural settings and researchers avoid manipulating the research outcomes a priori [17]. A move towards confirmability removes the issue from “the investigator’s characteristics” to “the characteristics of the data: Are they or are they not confirmable?” [17] According to Lincoln and Guba, confirmability is tightly linked with auditability, and they argue that research studies must establish an audit trail consisting of (e.g.) raw data, data reduction and analysis products, data reconstruction and synthesis products, process notes, materials relating to intentions and dispositions, and instrument development information [17]. Confirmability thus requires that the research design is constructed in such a way that the audit trail is preserved, and Lincoln and Guba also state that an actual audit must take place.

Similar thoughts to that of Lincoln and Guba can be found in several other qualitative methods textbooks. Examples include Miles and Huberman [19] who proposed a set of questions to ask of a qualitative study about objectivity. These questions cover to what extent the methods are described explicitly and in detail, whether there is a record of the study detailed enough to be considered an audit trail, whether it is possible to follow the sequence of data collection, processing and presentation, whether researcher assumptions, values and biases are made explicit, whether competing hypotheses are considered, and whether study data is retained and available for re-analysis by others. Additionally, Collins in his introduction to sociology research affirms that “if qualitative research is to deserve the label of “science” it should be conducted in such a way that it could be replicated *in principle*” [7]. Collins however does not only link replicability to method concerns, but also to the ability to generalise from the results - “as the significance broadens, there are more and more ways of checking” [7]. On a more practical level, confirmability seems analogous to accountability of (e.g.) service providers, since both concepts aim to verify that the researchers cf. service providers are “doing the right thing”. In security research,

the concept of accountability came into prominence with the introduction of the EU General Data Protection Regulation, where the ability to demonstrate that handling of sensitive personally identifiable information is performed in a compliant manner became an explicit requirement. An accountable organization must define what it does when it handles personal data, monitor how it acts, remedy any discrepancies between the definition of what should occur and what is actually occurring, and explain and justify any related action [16].

The concept of strong objectivity stems from feminist standpoint theory [11]. Standpoint theory points out that all knowledge arises in particular social situations with people with particular social positions, and thus is not value-free. The concept of strong objectivity brings the assumptions and agendas of the researchers into the research as part of what is investigated, acknowledging that these are not easily detected by individuals. It claims that by taking the perspectives of the marginalized or oppressed one can achieve more objective knowledge. Marginalised individuals are "outsiders within" [11] and are thus able to understand both their own position and that of the dominant culture.

Gaukroger additionally points out another possible understanding of objectivity, namely that "something is objective if it leads to conclusions which are universally accepted" [9]. Similarly, Robson [22] claims that 'objective' can be taken to refer to what multiple observers agree to as a phenomenon, in contrast to the subjective experience of the single individual. However, Gaukroger warns that "we should not assume that there is a correlation between degree of agreement and degree of objectivity" [9].

To sum up, objectivity can be understood in different ways. There is a need to consider for particular cases "what we want out of objectivity" [9] and how objectivity can be secured. In the remaining parts of this paper we look more closely at objectivity related to judgement about the security level, and in particular what level of security is "good enough". We make use of the understandings of objectivity as 1) procedures that allow one to decide between conflicting views, 2) accurate representations, and 3) freedom from prejudice and bias, and we touch upon the understanding of objectivity as universally accepted conclusions (see Table 1).

### 3 RESEARCH METHODOLOGY

Our claims about agile software security in this paper are based on involvement over several years with companies on the topic of software security in agile development. This includes interview studies involving about 20 public companies with the aim to identify practices and challenges in agile development [15, 28] and action research involving several companies, as part of the SoS-Agile research project [8]. In this project we have investigated how to meaningfully integrate software security into agile software development activities. The companies we have worked with are varied in their size, the type of software they develop and their organization, and include smaller development departments, distributed development teams, and larger development organisations. We have studied individual projects, as well as overall organizational approaches to software security in agile development. Our major involvement has been with three companies, and these have

**Table 1: Understandings of objectivity used in the examples**

Understanding of objectivity	Example from software security
Objectivity as a procedure that allows one to decide between conflicting views – with the scientific method and its focus on falsification as an example of such a procedure.	To what extent "we have good enough software security" is a falsifiable claim, i.e., whether one is able to identify cases of security levels that are too low as well as too high.
Objectivity as accurate representations, and objectivity as universally accepted conclusions.	The different roles/stakeholders involved in judgements about software security, and their varying viewpoints and understanding about software security.
Objectivity as freedom from prejudice and bias.	The prejudice and bias commonly found among security experts when approaching a development project.

been studied over several years. In addition to those, we have had shorter collaborations with five companies on more specific issues.

In action research, the aim is to merge theory and practice in such a way that real-world problems are solved by theory-informed actions in collaboration between researchers and practitioners [10]. In our research, the "action" has been the introduction of various security practices; threat modeling, static analysis tools, self management for security and security requirements work. To obtain a wide understanding of the transformation phenomenon, various data collection mechanisms have been applied, including observations, interviews, questionnaires and document analysis, and we have built a close relationship with the software companies. Building a close relationship is important in any action research study. Aspects of security work however makes it even more important, in our experience. This is due to the secrecy and sensitivity of the information and artefacts that are dealt with in the organization, but also that security requirements are mostly non-functional and not really the focus in the daily activities of software teams.

### 4 OBJECTIVITY CONCERNS IN JUDGEMENTS ABOUT "GOOD ENOUGH" SOFTWARE SECURITY

In the following we exemplify challenges in reaching objective evaluations of the level of software security, and in particular whether the level of software security is "good enough". We draw upon different understandings of objectivity, as shown in Table 1.

#### 4.1 Is "we have good enough security" a falsifiable claim?

There are a variety of ways one may go about evaluating the level of software security in a project. Still, judging whether the security level is too high or too low – currently and in the near future – is not straightforward. We illustrate this with some examples.

A naïve evaluation of the claim “we have good enough security” would be to consider whether the system experiences any security incidents; thus a system that is successfully attacked would not be satisfactorily secure. There is some merit to this, however, there is a need for ways to evaluate security that are less reactive. Results of code review, static code analysis and security testing offer a more proactive evaluation of security, and can provide some assertion that one is on the right track. This does not mean, though, that it is clear how much analysis and testing is enough. Additionally, all security issues identified using such approaches do not necessarily need to be addressed to achieve good enough security. Another seemingly straightforward way of addressing whether the software security is good enough, is to consider whether it meets legislative and customer requirements on security. However, customers often consider security as an implicit requirement that should be taken care of by the developers [2], and legislative requirements and their concrete implications for the project can be a case of debate and negotiation between different types of experts [28].

It has been argued that it may be easier to evaluate software security based on the processes performed, rather than the software itself [18]. In our research and interaction with software development companies we have often used the Building Security In Maturity Model (BSIMM) [18, 31] as a tool to help companies identify practices that they want to apply or improve [15]. All software security activities included in the BSIMM are activities that are performed in real companies. The BSIMM additionally identifies which activities are most commonly adopted by software companies, and, by extension, most companies would probably benefit from doing. However, companies are of different size and develop software with different security requirements. In our work with small and medium sized software development companies, we find that it is not trivial to know which activities to recommend to a software company, despite the knowledge of which activities are most common. A BSIMM evaluation would give you knowledge about what activities you do and don't do, but not whether you have adopted a set of practices that fits your needs. Additionally, doing a full scale BSIMM evaluation is costly. In our use of BSIMM as a research tool we have relied heavily on self-evaluation, and thus on the company's own understanding of their own practices. Interestingly, we have observed that in one company their overall BSIMM self-evaluation scores actually dropped after investing in several improvements in their software security practices. This was because they now had a better understanding of the implications of the BSIMM activities and the limitations of their own practices.

A mantra in most security work is that the approach should be risk based, meaning that one is aware of what the main risks are, and targets those in a strategic manner. Risk assessments should then ideally help software projects identify this “good enough” level of security. There exists a variety of methods for performing risk assessments related to information or software security, some highly detailed and quantitative in nature, others less formal and qualitative in nature. There is research showing that software projects often do not have a risk-based approach to software security [28]. Still, it is safe to assume that agile software projects would typically rely on qualitative risk analysis with expert evaluations for security risk assessments, as this is a cost-effective approach and can be done in a relatively short amount of time. Such analysis has been

critiqued for not measuring risk, but rather “human judgement about security risk” [12], and that though this judgement can be useful, it comes with its limitations. It is an open question how much effort needs to be put into a risk analysis for the result to be reliable. Note also that security is in many ways a moving target – new vulnerabilities and attacks can invalidate previous assumptions and thus demand a new risk assessment to be performed.

The kinds of security analysis introduced above would mainly be able to identify lacks in security. Indications of too much security would probably come in other forms, e.g. through loss in competitiveness and broken deadlines. These are however very indirect measures of the security level, and these problems may stem from sources not related to security as well. There are many approaches to evaluate cyber security investments, with the Return on Security Investments being one example [4]. In practice however, we find that companies we interact with only discuss this in informal terms.

Based on the above we would claim that though it may be possible to state after-the-fact that the security at some point was too low, it is very difficult to know if a project invests more than necessary on security, or if the same investments could be more efficiently used in a different way. Indicators may be introduced throughout, but this does not necessarily increase objectivity if not paired with proper judgment about what kind of decision support they provide.

## 4.2 Can you see something you don't have knowledge about?

In ongoing research on software security requirements and priorities in agile projects, we have used interviews as one of the data collection methods, and have among other things asked different actors in a software development project to what extent they believe they ended up with good enough security in the project; not too high or too low. In a project where we asked a security champion (SC) of a team, a product owner (PO) and a technical product owner (TPO) this question, they all agreed the security was good enough, but they had different explanations as to why that was the case.

The SC is a developer that is a regular part of the development team but has been assigned some responsibility for security. The SC thought security had been given the right priority. The project could have done more on security, but then the SC believed they would have had problems finishing. The SC had observed a raised security awareness in the development team compared to previous projects, and that this had impacted the software developed, thus the security level was not considered too low either.

The TPO has a background as a developer and software architect and brings his technical background into strategic discussions and priorities in the project. The TPO believed the level of security was about right, but that the sense of responsibility for security should be different so that everybody took responsibility for security without having to make a big process around it. The TPO believed security was important but did not have capacity to take this on as yet another task.

The PO is responsible for prioritising the requirements and represents the customer interests. In this project the PO had focus on following up the formal security requirements towards the customer to ensure the contractual obligations were met. The PO trusted the

TPO and the SC, was aware that the SC and TPO spent some time on security, and consequently trusted that the level of security was about right.

We bring out this example to illustrate that the security level and priorities are viewed differently based on the position and the competence of the one making the evaluation; in this case from a perspective of how security is perceived by the team (SC), the sense of responsibility (TPO) and trust in others (PO). The PO that does not have that deep technical knowledge, including on security, is not in a position to see security problems if not told about this, and is not aware of security issues that have come up along the way and have been decided upon by the team (while the SC and TPO are aware of this).

With such varying viewpoints, a relevant question is ‘What is an accurate representation?’ The common understanding is not necessarily the most objective one, as this is a highly specialized topic and objectivity in representations can require skills and training [9]. However, is it also possible that individuals with a lot of security awareness and knowledge may see “too many” security issues or at least more than what you, from a business perspective, would want to invest on fixing?

### 4.3 Is prejudice and bias a driver for software security assessments and research?

Security analysis and research is often done based on the assumption that the security work currently is not good enough and needs to be improved. This can be considered a form of prejudice and bias – the state of software security is, before any study has been performed, considered to be too low. Adding to this potential challenge is what has been characterised as a “disconnect between security and development” [29] stemming from these expert communities traditionally being isolated from each other. Thus, security experts commonly lack an adequate understanding of development [29].

The role and mindset of a security expert is often to identify problems; that lies in the nature of the field and the tasks. This could be represented by the auditor role. Security experts could however assume another role, that of the guide or the supervisor, providing support to developers and strengthening what they are already doing that is good. This requires another skill set [29] and possibly a more positive attitude. It is an open question to what extent the mindset of the security expert, it being that of auditor or support, affects how they assess the security level.

## 5 WAY FORWARD

The previous section exemplified challenges of making objective judgements of whether the security is good enough. In the following we suggest ways in which practitioners and security researchers can draw on the different understandings of objectivity to improve their judgements of security. We discuss the following strategies: including a variety of perspectives, building interactional expertise, and supporting confirmability.

In security analysis, it is quite common to aim to “think like an attacker”, e.g. as is done in threat modeling [25]. We would claim that bringing in more perspectives is one way of increasing accuracy of representation. Relevant perspectives include not only that of the attacker, but that of developers, operations, customers,

users, managers, etc. When taking the attacker’s perspective this is done as an exercise in thinking, but other types of actors may even be invited to take part in the analysis. Several security techniques support this kind of involvement; risk analysis can be done with a wide range of participants, games such as Protection Poker [30, 32] invite broad attendance though mainly from the development team, and techniques such as the Security Intention Meeting [27] aim to include the management level regularly in high-level security analysis and decisions. The cost of such involvement, however, needs to be taken into account.

In addition to improving accuracy of representation, bringing in representatives of different perspectives can potentially help flash out prejudices and bias of the researcher or security practitioner doing the security analysis. The concept of strong objectivity [11] points to this potential role of being “gazed back” at from the objects of study, and through this being able to gaze back at oneself, one’s own socially situated beliefs and practices, from a location further away from daily work [11]. The true effects of strong objectivity come with practices that are too extensive and thus out of the scope for the topic of this paper. Still, one may likely experience some of these effects simply by including and truly listen to other perspectives on the security level. Experiences from using the concept of strong objectivity in a transdisciplinary research project [23] point to benefits of being open and transparent about own positions and standpoints; “there *always* exist value judgements in science. Reaching objectivity requires not only making these transparent and accessible, but also necessitates submitting those judgements to an open and rational debate” [23]. This goes beyond just engaging different stakeholders and includes such things as addressing power imbalances.

In literature, the PO role has been found to commonly limit the priority given to security [1, 26]. The business case for security is often considered unclear [26] while the push for functionality is strong, and this results in less focus on security [1, 26, 28]. In our experience with companies, we see that the PO role can act as a hindrance for security in many cases. Additionally, we observe that POs often have limited competence about software security. Building security competence at the PO level could be a way to increase the PO’s ability to make good judgements related to security priorities. This is however not a one way street. There is a need for security experts, as well as security concerned developers, to understand the perspectives of the PO so that these roles can have fruitful discussions about security and project priorities. To cite Sandhu: “We are completely clueless about what is good enough. [...] Business people cannot tell us because they don’t understand security and security people cannot tell us because they don’t understand business. We must close this divide” [24].

We exemplify this need by addressing an underlying assumption in this paper, namely that too much security will lead to drawbacks such as increased cost, reduced usability, etc. Though this is commonly considered to be the case, it is not necessarily always true. The Privacy by Design [6] initiative does in one of its principles encourage the move away from zero-sum thinking about privacy, to positive-sum, looking for win-win solutions that is good for privacy as well as other goals of the system. In the same way, if the thinking about security is mainly that it hampers other system goals, one may miss solution alternatives where security can help achieving



other types of goals in the system as well. Having security experts with a better understanding of the project goals is one possible step towards a kind of thinking that may lead to positive-sum solutions. The need for competence lies at the level of interactional expertise [7], that is, there is a need for being able to understand each other, speaking the language, but not being able to perform or directly contribute to each others tasks. Meeting and talking together can help build the necessary trust and understanding [7].

This paper has informally discussed several types of indicators to approach an evaluation of what is good enough, including the time spent on security, security testing results, what other companies do (BSIMM), what some experts assess to be the need (risk analysis), the security awareness in the development team, etc. More research is needed to know to what extent there is a good correlation between any of these indicators and the security of the final system. We do not in this paper make any claims as to what types of security analysis methods would best support objective security evaluations, apart from our recommendation to include different viewpoints in the analysis. More research is needed in order to make such claims. However, we draw on the concept of confirmability in recommending that the judgements as well as the reasons behind any judgements are kept so that assumptions and decisions can be revisited at a later stage. This is important in case there is a security incident, but also in order to deal with changing threat landscapes and project goals. Note however that though this is to some extent in conflict with the agile manifesto and its emphasis on "[w]orking software over comprehensive documentation" [3], agile is not about no documentation, and it is possible to document these types of issues as part of e.g. the software's structure, commit messages, unit tests and comments.

## 6 CONCLUSION

This paper has used theory on objectivity to see how it can improve both researchers' and practitioners' assessment of what is good enough when it comes to software security. More research is needed in order to provide agile-friendly and concrete method support on achieving objective judgements about what is "good enough". As a way forward, this paper suggests researching and adopting practices that include different perspectives, supports the building of interactive expertise among key actors, and that support confirmability by documenting judgements about security and their rationale.

## ACKNOWLEDGMENTS

This work was supported by the SoS-Agile: Science of Security in Agile Software Development project, funded by the Research Council of Norway (grant number 247678, <https://www.sintef.no/en/digital/sos-agile/>). Thanks to Prof. Jonathan Knowles and Prof. Colin Boyd for commenting on this paper at various stages.

## REFERENCES

- [1] Wasim Alsaqaf, Maya Daneva, and Roel Wieringa. 2017. Quality requirements in large-scale distributed agile projects—a systematic literature review. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 219–234.
- [2] S. Bartsch. 2011. Practitioners' Perspectives on Security in Agile Development. In *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*. IEEE, 479–484. <https://doi.org/10.1109/ARES.2011.82>
- [3] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. 2001. Manifesto for agile software development. *Online at <https://www.agilemanifesto.org>* (2001).
- [4] Stefan Beissel et al. 2016. *Cybersecurity investments*. Springer.
- [5] Konstantin Beznosov. 2003. Extreme security engineering: On employing XP practices to achieve 'good enough security' without defining it. In *First ACM Workshop on Business Driven Security Engineering (BizSec)*. Fairfax, VA, CiteSeer.
- [6] Ann Cavoukian et al. 2009. Privacy by design: The 7 foundational principles. *Information and Privacy Commissioner of Ontario, Canada* 5 (2009).
- [7] Harry Collins. 2019. *Forms of Life: The Method and Meaning of Sociology*. MIT Press.
- [8] Daniela S. Cruzes, Martin G. Jaatun, and Tosin D. Oyetoyan. 2018. Challenges and Approaches of Performing Canonical Action Research in Software Security: Research Paper. In *Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security (HoTSoS '18)*. ACM, New York, NY, USA, Article 8, 11 pages. <https://doi.org/10.1145/3190619.3190634>
- [9] Stephen Gaukroger. 2012. *Objectivity: A very short introduction*. Oxford University Press.
- [10] Daryydd J Greenwood and Morten Levin. 2006. *Introduction to action research: Social research for social change*. SAGE publications.
- [11] Sandra Harding. 1991. Strong objectivity<sup>3</sup> and socially situated knowledge. *Women science* (1991), 138–163.
- [12] Lance Hayden. 2010. *IT security metrics: A practical framework for measuring security & protecting data*. Vol. 396. McGraw Hill New York.
- [13] George Hurlburt. 2016. " Good Enough" Security: The Best We'll Ever Have. *Computer* 49, 7 (2016), 98–101.
- [14] ISO. 2013. Information technology – Security techniques – Information security management systems – Requirements. ISO/IEC Standard 27001:2013. <https://www.iso.org/standard/54534.html>
- [15] Martin Gilje Jaatun, Daniela S. Cruzes, Karin Bernsmed, Inger Anne Tøndel, and Lillian Røstad. 2015. Software Security Maturity in Public Organisations. In *Information Security*, Javier Lopez and Chris J. Mitchell (Eds.), Lecture Notes in Computer Science, Vol. 9290. Springer International Publishing, 120–138.
- [16] Martin Gilje Jaatun, Siani Pearson, Frédéric Gittler, Ronald Leenes, and Maartje Neezen. 2016. Enhancing Accountability in the Cloud. *International Journal of Information Management* (2016). <https://doi.org/10.1016/j.ijinfomgt.2016.03.004>
- [17] Yvonna S Lincoln and Egon G Guba. 1985. *Naturalistic inquiry*. Sage Publications Inc.
- [18] Gary McGraw, Sammy Miguez, and Jacob West. 2018. *BSIMM 9*. Technical Report. Synopsys, Inc.
- [19] Matthew B. Miles and A. Michael Huberman. 1994. *Qualitative data analysis: An expanded sourcebook* (2nd ed.). Sage.
- [20] Karl R Popper. 1972. *Objective Knowledge: An Evolutionary Approach*. Oxford University Press, Chapter The bucket and the searchlight: Two theories of knowledge.
- [21] David B Resnik et al. 2011. What is ethics in research & why is it important. *National Institute of Environmental Health Sciences* 1, 10 (2011), 49–70.
- [22] Colin Robson. 2011. *Real World Research* (3 ed.). John Wiley & Sons.
- [23] Judith Rosendahl, Mathias A Zanella, Stephan Rist, and Jes Weigelt. 2015. Scientists' situated knowledge: Strong objectivity in transdisciplinary. *Futures* 65 (2015), 17–27.
- [24] Ravi Sandhu. 2003. Good-enough security. *IEEE Internet Computing* 7, 1 (2003), 66–68.
- [25] Adam Shostack. 2014. *Threat Modeling: Designing for Security*. Wiley.
- [26] Evenynke Terpstra, Maya Daneva, and Chong Wang. 2017. Agile Practitioners' Understanding of Security Requirements: Insights from a Grounded Theory Analysis. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*. IEEE, 439–442.
- [27] Inger Anne Tøndel, Daniela Soares Cruzes, Martin Gilje Jaatun, and Kalle Rindell. 2019. The Security Intention Meeting Series as a way to increase visibility of software security decisions in agile development projects. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*. ACM, 59.
- [28] Inger Anne Tøndel, Martin Gilje Jaatun, Daniela Soares Cruzes, and Nils Brede Moe. 2017. Risk Centric Activities in Secure Software Development in Public Organisations. *International Journal of Secure Software Engineering (IJSSSE)* 8, 4 (2017), 1–30.
- [29] K. R. van Wyk and G. McGraw. 2005. Bridging the gap between software development and information security. *IEEE Security & Privacy* 3, 5 (2005), 75–79.
- [30] Laurie Williams, Michael Gegick, and Andrew Meneely. 2009. Protection poker: Structuring software security risk assessment and knowledge transfer. In *International Symposium on Engineering Secure Software and Systems*. Springer, 122–134.
- [31] Laurie Williams, Gary McGraw, and Sammy Miguez. 2018. Engineering Security Vulnerability Prevention, Detection, and Response. *IEEE Software* 35, 5 (2018), 76–80.
- [32] Laurie Williams, Andrew Meneely, and Grant Shipley. 2010. Protection poker: The new software security game. *IEEE Security and Privacy* 8, 3 (2010), 14–20.

## **Paper G: ‘Influencing the security prioritisation of an agile software development project’**

Included is the published material [33], following the Creative Commons Attribution 4.0 International (CC BY 4.0) licensing arrangement used by Elsevier.



**G**



Contents lists available at ScienceDirect

Computers &amp; Security

journal homepage: [www.elsevier.com/locate/cose](http://www.elsevier.com/locate/cose)

# Influencing the security prioritisation of an agile software development project

Inger Anne Tøndel<sup>a,\*</sup>, Daniela Soares Cruzes<sup>a,b</sup>, Martin Gilje Jaatun<sup>b</sup>, Guttorm Sindre<sup>a</sup>

<sup>a</sup>Department of Computer Science, Norwegian University of Science and Technology (NTNU), Sem Sælandsvei 9, Gløshaugen, Trondheim 7034, Norway  
<sup>b</sup>SINTEF Digital, Strindvegen 4, Trondheim 7034, Norway

## ARTICLE INFO

### Article history:

Received 22 December 2021

Revised 19 April 2022

Accepted 23 April 2022

Available online 25 April 2022

### Keywords:

Software security

Agile software development

Case study

Security priority

Security requirements

## 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 – prescribed and emerging – where we hypothesise that an emerging approach can be more relevant for SMEs doing ASD, and that this can impact how such companies should consider software security maturity.

© 2022 The Authors. Published by Elsevier Ltd.

This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>)

## 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; Jarzabowicz et al., 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; Jarzabowicz et al., 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;

\* Corresponding author.

E-mail address: [inger.anne.tondel@ntnu.no](mailto:inger.anne.tondel@ntnu.no) (I.A. Tøndel).

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; Jarzbowicz 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) (<https://cwe.mitre.org/>) and the Common Attack Pattern Enumeration and Classification (CAPEC) (<https://capec.mitre.org/>)) all illustrate the broadness and the extensiveness of the software security work.

In this article we consider the concept *security prioritisation* to include prioritisation amongst 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; Jarzbowicz 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 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 is 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 amongst 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-orientated 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 amongst 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 amongst 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 Fig. 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 collect 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.

### 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). Fig. 2 gives an overview of the analysis process for arriving at the influence categories (RQ1).

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,

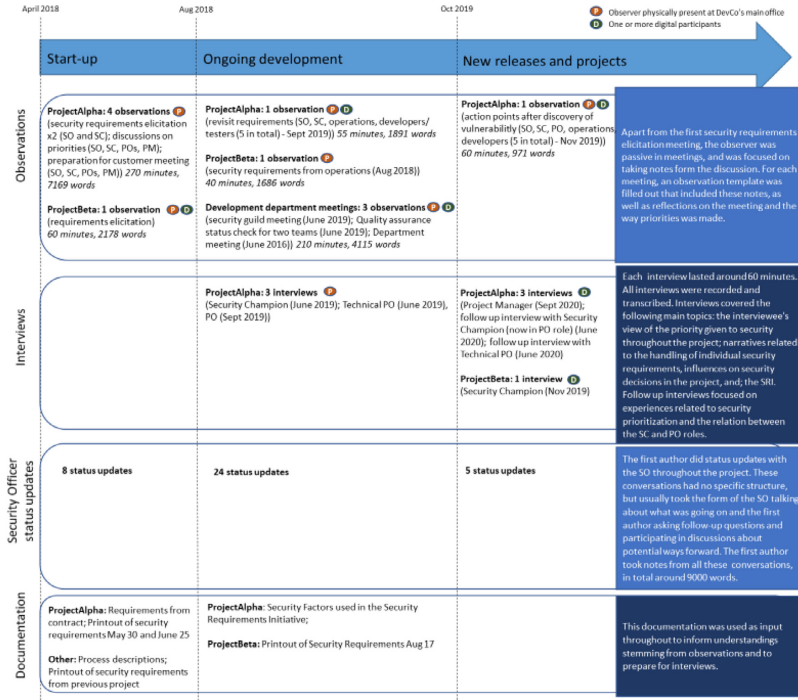


Fig. 1. Overview of data collection (SO = Security Officer, SC = Security Champion, PM = Project Manager, PO = Product Owner, SRI = Security Requirements Initiative).

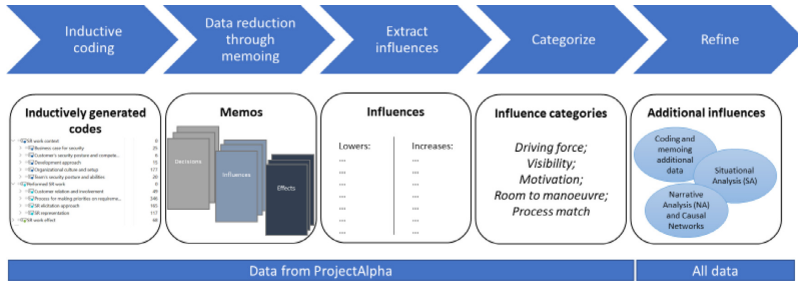


Fig. 2. Analysis process for arriving at influences and influence categories.

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 rele-

vant 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. Fig. 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 amongst 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

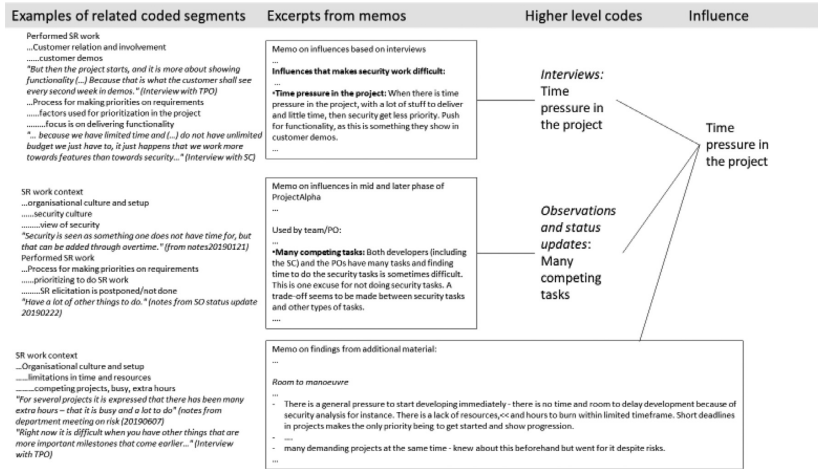


Fig. 3. Example showing part of the link between one identified influence and coded segments, demonstrating how the identified influences are related to memos that are linked to coded segments from the data material that are ordered according to categories from Tøndel and Jaatun (2020).

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 categorized 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. Fig. 3 demonstrates how the influences we identified through this analysis process can be traced back to coded segments.

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 craftsmanship 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 evolution 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 (Fig. 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.



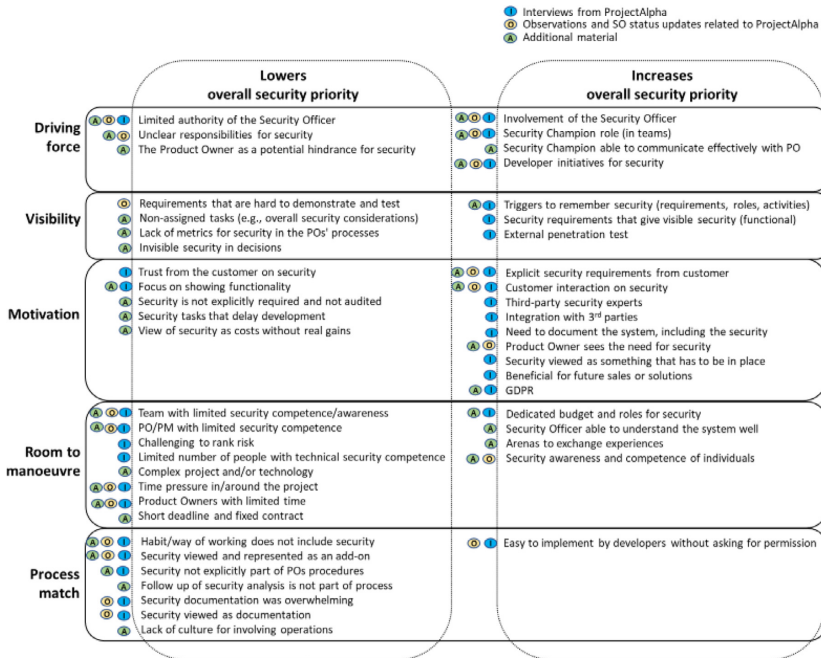


Fig. 4. Conceptual model of influences on security priority (PM = Project Manager, PO = Product Owner).

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. Fig. 5 gives an overview of the evolution of the Security Requirements Initiative that the Security Officer initiated.

4.1. Driving force

There were several roles that had a potential influence on the priority given to security in this project. Fig. 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.

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 Project-Beta, 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 amongst 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 develop-





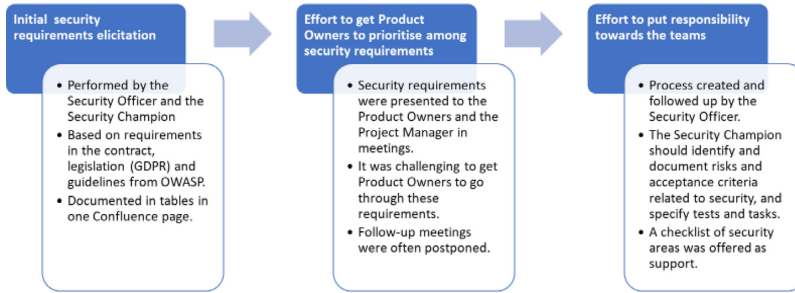


Fig. 5. Overview of the stages of the Security Requirements Initiative.

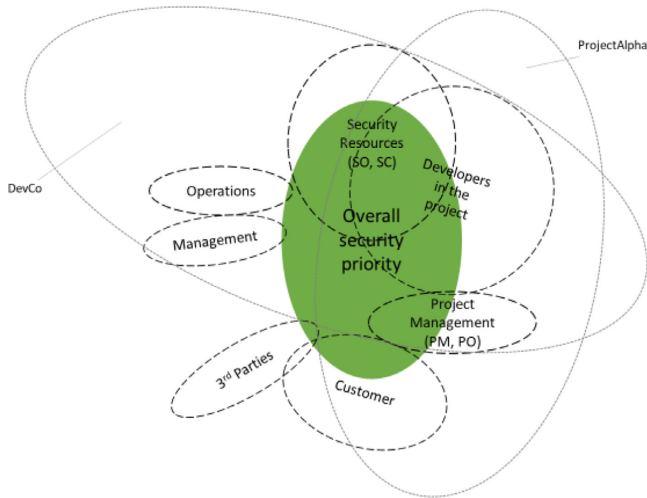


Fig. 6. Social worlds/arenas map for ProjectAlpha (SO = Security Officer, SC = Security Champion, PM = Project Manager, PO = Product Owner).

ers, 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. Fig. 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 activ-

ities 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.

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*

**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 (Fig. 4)
Security Resources: Security Officer	<ul style="list-style-type: none"> <li>Initiated and followed up the Security Requirements Initiative, arranged meetings and poked individuals about security tasks.</li> <li>Later, the Security Officer left the company and a drop in security focus was observed.</li> <li>Had to battle issues with formal authority, as the Product Owners, the Project Manager, and operations were not in the development department.</li> <li>It was not scalable to be personally involved in all follow-ups on security.</li> </ul>	<ul style="list-style-type: none"> <li>+ Involvement of the Security Officer</li> <li>- Limited authority of the Security Officer</li> </ul>
Security Resources: Security Champion	<ul style="list-style-type: none"> <li>Supported continuity of security attention in the development teams.</li> <li>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.</li> </ul>	<ul style="list-style-type: none"> <li>+ Security Champion role in teams</li> <li>+ Security Champion able to communicate effectively with Product Owner</li> <li>+ Developer initiatives for security</li> </ul>
Developers in the project	<ul style="list-style-type: none"> <li>Several developer initiatives for security came in addition to the explicit security requirements from the customer.</li> <li>Developers did the coding and made choices on which security mechanisms to apply.</li> <li>Influence on the project management roles because of technical competence.</li> </ul>	<ul style="list-style-type: none"> <li>- Unclear responsibilities for security</li> <li>- The Product Owner as a potential hindrance for security</li> </ul>
Project Management (Project Manager, Product Owners)	<ul style="list-style-type: none"> <li>Influence through prioritisation and budget, but largely delegated prioritisation to the security resources, and to some extent to the developers.</li> <li>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.</li> <li>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.</li> </ul>	<ul style="list-style-type: none"> <li>- Unclear responsibilities for security</li> <li>- The Product Owner as a potential hindrance for security</li> </ul>
Customer	<ul style="list-style-type: none"> <li>Indirect influence through the explicit security requirements.</li> <li>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.</li> </ul>	(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) Management	<ul style="list-style-type: none"> <li>Technological solutions from 3rd parties could support (or not support) the security requirements, thus indirectly allowing for (or hampering) their prioritisation.</li> <li>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.</li> <li>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.</li> </ul>	(Indirect influence, thus not included as driving force)  (No clear examples of how this influence played out, thus not included as driving force)
Operations	<ul style="list-style-type: none"> <li>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.</li> </ul>	(Potential driving force, but not much active in the project)

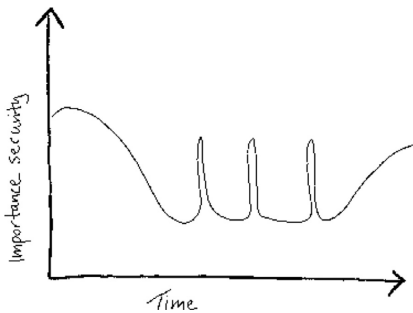


Fig. 7. Typical timeline drawn in interviews.

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 compla-

cent. 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.

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.



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

Stage in timeline	How this was influenced by visibility	Influence in conceptual model (Fig. 4)
Initial high security attention	<ul style="list-style-type: none"> <li>• Security analysis was required from the customer in the beginning of the project.</li> <li>• The Security Requirements Initiative pushed for the security priorities being made early in the project.</li> </ul>	+ Triggers to remember security
Drop in security attention	<ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> <li>• 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.</li> </ul>	<ul style="list-style-type: none"> <li>- Requirements that are hard to demonstrate and test</li> <li>- Non-assigned tasks</li> <li>- Lack of metrics for security in the Product Owners' processes</li> <li>- Invisible security in decisions</li> </ul>
Some ongoing attention, despite the drop	<ul style="list-style-type: none"> <li>• 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).</li> </ul>	+ Triggers to remember security
Spikes	<ul style="list-style-type: none"> <li>• In iterations where security requirements were to be implemented, security was more visible.</li> <li>• 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.</li> <li>• Customer interaction on security (e.g., presentations, questions) put security on the agenda of Project Managers and Product Owners.</li> <li>• Performing an external penetration test increased visibility of security towards the Project Manager and Product Owner roles.</li> </ul>	<ul style="list-style-type: none"> <li>+ Triggers to remember security</li> <li>+ Security requirements that give visible security</li> <li>+ External penetration test</li> </ul>
Increase towards release	<ul style="list-style-type: none"> <li>• Security became visible as part of a need to check whether all the explicit security requirements from the contract were fulfilled.</li> </ul>	+ Triggers to remember security

#### 4.4. Room to manoeuvre

The room to manoeuvre is dependant 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.

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.

The Security Officer experienced several challenges when it came to integrating the Security Requirements Initiative into the processes of DevCo. As seen in Fig. 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, Fig. 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,

**Table 3**

Overview of the role of influences related to motivation on the security priority timeline in Fig. 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 (Fig. 4)
Initial high security attention	<ul style="list-style-type: none"> <li>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.</li> <li>Presentation of the security analysis to the customer led to a motivation to do a good job on this analysis.</li> </ul>	<ul style="list-style-type: none"> <li>+ Explicit security requirements from customer</li> <li>+ Customer interaction on security</li> </ul>
Drop in security attention	<ul style="list-style-type: none"> <li>They experienced perceived trust from the customer on security. Throughout, customer meetings emphasised the showing of functionality.</li> <li>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.</li> <li>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 security of an already working solution was low, and the approach to security became more reactive.</li> </ul>	<ul style="list-style-type: none"> <li>- Trust from the customer on security</li> <li>- Focus on showing functionality</li> <li>- Security is not explicitly required and not audited</li> <li>- Security tasks that delay development</li> <li>- View of security as costs without real gains</li> </ul>
Some ongoing attention, despite the drop	<ul style="list-style-type: none"> <li>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.</li> <li>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."</li> <li>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.</li> <li>GDPR represented a push and motivation for security, and requirements related to GDPR came up in the observed meetings.</li> <li>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).</li> </ul>	<ul style="list-style-type: none"> <li>+ Third-party security experts</li> <li>+ Integration with 3rd parties</li> <li>+ Product Owner sees the need for security</li> <li>+ Security viewed as something that has to be in place</li> <li>+ Beneficial for future sales or solutions</li> <li>+ GDPR</li> </ul>
Spikes	<ul style="list-style-type: none"> <li>Customer interaction on security (e.g., presentations, questions) increased the motivation to do a good job on security.</li> </ul>	<ul style="list-style-type: none"> <li>+ Explicit security requirements from customer</li> <li>+ Customer interaction on security</li> </ul>
Increase towards release	<ul style="list-style-type: none"> <li>There was a need to check that all the promised security had been delivered. Meetings were arranged to discuss this.</li> <li>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.</li> </ul>	<ul style="list-style-type: none"> <li>+ Explicit security requirements from customer</li> <li>+ Need to document the system, including the security</li> </ul>

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.

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*



**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 (Fig. 4)
Time and budget	<ul style="list-style-type: none"> <li>The Security Champion and Security Officer roles had pre-allocated time to work on security tasks.</li> <li>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.</li> <li>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.</li> <li>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 <i>"surprised that I am not performing better than those that were Product Owners while I was Security Champion and developer"</i>.</li> </ul>	<ul style="list-style-type: none"> <li>+ Dedicated budget and roles for security</li> <li>- Time pressure in/around the project</li> <li>- Product Owner with limited time</li> <li>- Short deadline and fixed contract</li> </ul>
Competence	<ul style="list-style-type: none"> <li>The influence of the Security Officer was dependant 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.</li> <li>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 amongst all Security Champions and through the Security Officer being involved in a broad set of projects.</li> <li>The security in the development projects was highly dependant on the security awareness and competence of the developers, as they were creating their own security initiatives. Security competence and awareness varied greatly amongst the development teams of DevCo, and amongst Product Owners.</li> <li>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.</li> <li>With few individuals with security competence, these became a bottleneck.</li> <li>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.</li> </ul>	<ul style="list-style-type: none"> <li>+ Security Officer able to understand the system well</li> <li>+ Arenas to exchange experiences</li> <li>+ Security awareness and competence of individuals</li> <li>- Team with limited security competence/awareness</li> <li>- Product Owner / Project Manager with limited security competence</li> <li>- Challenging to rank risk</li> <li>- Limited number of people with technical security competence</li> <li>- Complex project and/or technology</li> </ul>

**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 (Fig. 4)
Product Owners' process	<ul style="list-style-type: none"> <li>Security was viewed and represented as an add-on having separate documents in the tender, and separate Confluence pages and procedures.</li> <li>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.</li> <li>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.</li> <li>Security activities were seen as "developers' job".</li> </ul>	<ul style="list-style-type: none"> <li>- Security viewed and represented as an add-on</li> <li>- Security not explicitly part of Product Owners' procedures</li> <li>- Follow up of security analysis is not part of the process</li> </ul>
Development process	<ul style="list-style-type: none"> <li>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.</li> <li>Security was to some extent viewed as documentation and talked about as <i>"spending a day to document the security around that function"</i>. This documentation was difficult to integrate with an agile way of working and was considered to <i>"use up lots of time"</i>.</li> <li>Teams already found themselves drowning in detail in Jira and security documentation just added to this already existing overload of information.</li> </ul>	<ul style="list-style-type: none"> <li>+ Easy to implement by developers without asking for permission</li> <li>- Security viewed as documentation</li> <li>- Security documentation was overwhelming</li> </ul>
Culture	<ul style="list-style-type: none"> <li>Some interviewees pointed to the role of habit, as they were generally not used to working with security this systematically.</li> <li>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.</li> </ul>	<ul style="list-style-type: none"> <li>- Habit/way of working does not include security</li> <li>- Lack of culture for involving operations</li> </ul>

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.



Fig. 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.

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.

**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 *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



**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 amongst 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) (+) Integrate security requirements (S1–3, 6–8, 17)
Process match	(-) Late security requirements (C9). (-) Planning sessions without quality stakeholders (C10). (-) Organisational structure (C21). (-) Limitations of agile (C19) and its implementation (C16). (+) Integrate security requirements (S1, 3–6) (+) Include security in estimates (S2)	(-) Contractors responsible – limited follow up. (-) Agile development. (-) CISO not able to follow up (+) New product.	(+) Integrate security activities (S9–10, 12, 15, 40, 42) (+) 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	

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 (van der 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 pillar 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







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

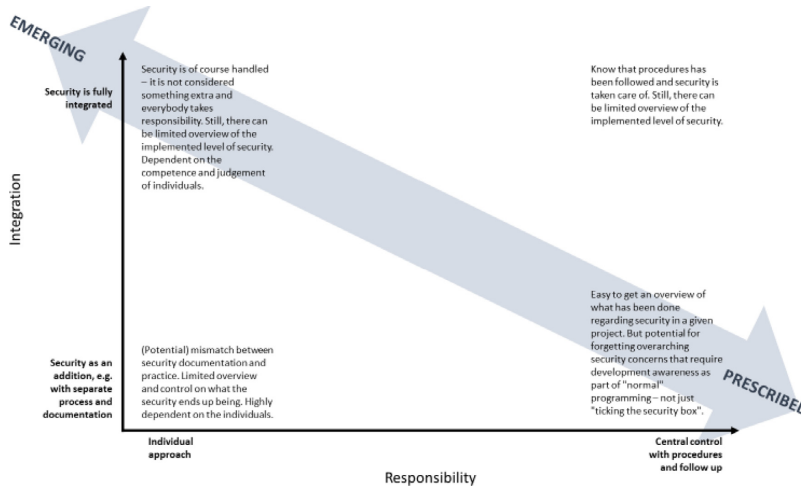


Fig. 10. Positional Map showing positions that can be taken related to placement of responsibility for security and level of integration of security into development. The emphasis is on potential consequences of the various positions for security.

been identified by considering Tables 1–5 as well as the experiences made by the Security Officer, extracting the key takeaways. Fig. 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 an emerging or prescribed approach to software security, is related to Fig. 10. This figure is a more general form of the Positional Map already presented in Fig. 8, and we use it to provide our understanding of what certain positions in the map would entail. In an *emerging* approach, security is fully

integrated with the way of working and everybody takes responsibility for security (upper-left). In a *prescribed* 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 emerging and prescribed 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 prescribed and emerging, these approaches are to some ex-





tent 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.

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 Fig. 10, that an emerging 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 orientated, 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 Fig. 10, it is possible to opt for a more emerging 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 Fig. 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 an emerging 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 hypothesize 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 prescribed or emerging 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 prescribed vs. emerging security approaches as inspiration to conduct studies aimed towards making recommendations for when companies should select a prescribed vs. emerging approach to security. Similarly, future studies could be aimed towards providing companies with recommendations on how to gradually build successful prescribed and emerging security initiatives in various contexts, as well as when and how to combine features of prescribed and emerging approaches successfully.

### Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

### CRedit authorship contribution statement

**Inger Anne Tøndel:** Conceptualization, Methodology, Formal analysis, Investigation, Writing – original draft, Visualization. **Daniela Soares Cruzes:** Conceptualization, Methodology, Writing – review & editing, Funding acquisition, Resources. **Martin Gilje Jaatun:** Writing – review & editing, Supervision. **Guttorm Sindre:** Writing – review & editing, Supervision.

### Acknowledgments

This research was funded by the [Research Council of Norway](#), Grant No. 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., Grünbacher, P., Perini, A., 2017. Quality requirements in large-scale distributed agile projects—a systematic literature review. In: *Requirements Engineering: Foundation for Software Quality*, 10153. Springer, Cham, pp. 219–234. doi:10.1007/978-3-319-54045-0\_17 REFSQ 2017.
- Alsaqaf, W., Daneva, M., Wieringa, R., 2019. Quality requirements challenges in the context of large-scale distributed agile: an empirical study. *Inf. Softw. Technol.* 110, 39–55. doi:10.1016/j.infsof.2019.01.009.
- Antukh, A., 2017. *Security Champions Playbook*. OWASP. <https://github.com/Cordis/security-champions-playbook/tree/master/Security%20Playbook>.
- Ashenden, D., Lawrence, D., 2016. Security dialogues: building better relationships between security and business. *IEEE Secur. Priv.* 14 (3), 82–87. doi: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. In: *Proceedings of the International Working Conference on Requirements Engineering* doi:10.1007/978-3-642-19858-8\_18. Foundation for Software Quality, Springer.
- Baldassarre, M.T., Barletta, V.S., Caivano, D., Piccinno, A., 2021. Integrating Security and Privacy in HCD-Scrum. In: *Proceedings of the 14th Biannual Conference of the Italian SICCHI Chapter*. Bolzano, Italy, p. 37. doi:10.1145/3464385.3464746.
- Bartsch, S., 2011. Practitioners' perspectives on security in agile development. In: *Proceedings of the Sixth International Conference on Availability, Reliability and Security (ARES)*, pp. 479–484. doi:10.1109/ARES.2011.82.
- K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries (2001) Manifesto for agile software development. <https://agilemanifesto.org/>
- Behutiye, W., Karhapää, P., López, L., Burgués, X., Martínez-Fernández, S., Vollmer, A.M., Rodríguez, P., Franch, X., Oivo, M., 2020. Management of quality requirements in agile and rapid software development: a systematic mapping study. *Inf. Softw. Technol.* 123, 106225. doi:10.1016/j.infsof.2019.106225.
- Blaine, J.D., Cleland-Huang, J., 2008. Software quality requirements: how to balance competing priorities. *IEEE Softw.* 25 (2), 22–24. doi:10.1109/MS.2008.46.
- Chowdhury, N.H., Adam, M.T.P., Teubner, T., 2020. Time pressure in human cybersecurity behavior: theoretical framework and countermeasures. *Comput. Secur.* 97, 101931. doi:10.1016/j.cose.2020.101931.
- Clarke, A.E., Frieze, C., Washburn, R., 2016. *Situational Analysis in Practice: Mapping Research with Grounded Theory*. Routledge.
- Cruzes, D.S., Johansen, E.A., 2021. Building an ambidextrous software security initiative. In: Manuel, M., Jorge Marx, G., Rory, V.O.C., Alena, B. (Eds.), *Balancing Agile and Disciplined Engineering and Management Approaches for IT Services and Software Products*. IGI Global, pp. 167–188. doi: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? In: *Proceedings of the IEEE 1st International Workshop on Quality Requirements in Agile Projects (QuaRAP)* doi:10.1109/QuaRAP.2018.00008.
- Eisenhardt, K.M., 1989. Building theories from case study research. *Acad. Manag. Rev.* 4 (14), 532–550. doi:10.5465/amb.1989.4308385.
- Freire, S., Rios, N., Gutierrez, B., Torres, D., Mendonça, M., Izurieta, C., Seaman, C., Spinola, R.O., 2020. Surveying software practitioners on technical debt payment practices and reasons for not paying off debt items. In: *Proceedings of the Evaluation and Assessment in Software Engineering (EASE)*, pp. 210–219. doi:10.1145/3383219.3383241.
- Inayat, I., Salim, S.S., Marczak, S., Daneva, M., Shamsirband, S., 2015. A systematic literature review on agile requirements engineering practices and challenges. *Comput. Hum. Behav.* 51, 915–929. doi:10.1016/j.chb.2014.10.046.
- Ionita, D., Cvd, V., Ikkink, H.J.K., Neven, E., Daneva, M., Kuipers, M., 2019. Towards risk-driven security requirements management in agile software development. In: *Proceedings of the International Conference on Advanced Information Systems Engineering* doi:10.1007/978-3-030-21257-1\_12, Springer.
- Jaatun, M.G., 2017. *The building security in maturity model as a research tool*. Empirical Research for Software Security: Foundations and Experience.

- Jaatun, M.G., Cruzes, D.S., 2021. Care and feeding of your security champion. In: *Proceedings of the International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, IEEE doi:10.1109/CyberSA52016.2021.9478254.
- Jaatun, M.G., Cruzes, D.S., Bernsmed, K., Tøndel, I.A., Røstad, L., Lopez, J., Mitchell, C., 2015. Software security maturity in public organisations. In: *Information Security, ISC 2015*, 9290. Springer, Cham, pp. 120–138. doi:10.1007/978-3-319-23318-5\_7.
- Jarżebowicz, A., Weichbroth, P., Przybyłek, A., Miler, J., Poth, A., Riel, A., 2021. A Systematic literature review on implementing non-functional requirements in agile software development: issues and facilitating practices. In: *Lean and Agile Software Development*, IASD 2021, 408. Springer, Cham, pp. 91–110. doi: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. *Empir. Softw. Eng.* 26, 28. doi: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. *Int. J. Softw. Eng. Appl.* 7, 49–68 IJSEAI.
- Koç, G., Aydos, M., 2017. Trustworthy scrum: development of secure software with scrum. In: *Proceedings of the 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. In: *Proceedings of the ACM on Human-Computer Interaction - CSCW* doi:10.1145/3274361.
- Loser, K.U., Degeling, M., Kimppa, K., Whitehouse, D., Kausela, T., Phahlamohaka, J., 2014. Security and privacy as hygiene factors of developer behavior in small and agile teams. In: *ICT and Society*, 431. Springer, Berlin, Heidelberg, pp. 255–265. doi:10.1007/978-3-662-44208-1\_21 HCC 2014.
- Maxwell, J.A., 2013. *Qualitative Research Design: an Interactive Approach*. Applied Social Research Methods 41 Sage publications.
- McGraw, G., 2006. *Software Security: Building Security in. Software Security: Building Security in*. Addison-Wesley Professional.
- Migues, S., Erikkhman, E., Ewers, J., Nassery, K., (2021) BSIMM12 2021 Found. Report. Synopsis. <https://www.bsimm.com/content/dam/bsimm/reports/bsimm12-foundations.pdf>
- Miles, M.B., Huberman, A.M., 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. *J. Syst. Softw.* 149, 217–233. doi: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. *Empir. Softw. Eng.* 26 (2), 1–29. doi:10.1007/s10664-020-09896-7.
- Oueslati, H., Rahman, M.M., Ib, O., 2015. Literature Review of the challenges of developing secure software using the agile approach. In: *Proceedings of the 10th International Conference on Availability, Reliability and Security*, pp. 540–547. doi:10.1109/ares.2015.69 24-27 Aug, 2015.
- Palombo, H., Tabari, A.Z., Lende, D., Ligatti, J., Ou, X., 2020. An ethnographic understanding of software (in) security and a co-creation model to improve secure software development. In: *Proceedings of the Sixteenth Symposium on Usable Privacy and Security SOUPS 2020*.
- Pelrine, J., 2011. On understanding software agility: a social complexity point of view. *Emerg. Complex. Organ.* 13, 26–37.
- Pohl, C., Hof, H.J., 2015. Secure scrum: development of secure software with scrum. *arXiv Prepr. arXiv:1507.02992*.
- Poller, A., Kocksch, L., Türlpe, S., Opp, F.A., Kinder-Kurlanda, K., 2017. Can security become a routine?: a study of organizational change in an agile software development group. In: *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. Portland, Oregon, USA, pp. 2489–2503. doi:10.1145/2998181.2998191.
- Riessman, C.K., 2008. *Narrative Methods for the Human Sciences*. Sage.
- Rindell, K., Hyrynsalmi, S., Leppänen, V., 2015. Securing scrum for VAHITI. In: *Proceedings of the 14th Symposium on Programming Languages and Software Tools*.
- Rindell, K., Hyrynsalmi, S., Leppänen, V., 2017. Busting a myth: review of agile security engineering methods. In: *ARES '17: Proceedings of the 12th International Conference on Availability, Reliability and Security*, pp. 1–10. doi:10.1145/3098954.3103170.
- Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. *Empir. Softw. Eng.* 2 (14), 131–164. doi: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* doi:10.36227/techrxiv.14141027.v1.
- Terpstra, E., Daneva, M., Wang, C., 2017. Agile practitioners' understanding of security requirements: insights from a grounded theory analysis. In: *Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pp. 439–442. doi:10.1109/REW.2017.54.
- Thomas, T.W., Tabassum, M., Chu, B., Lipford, H., 2018. Security during application development: an application security expert perspective. In: *CHI '18: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* doi: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. In: *Proceedings of the USENIX Symposium on Usable Privacy and Security (SOUPS) 2021*.

- Türpe, S., 2017. The trouble with security requirements. In: Proceedings of the IEEE 25th International Requirements Engineering Conference (RE). IEEE doi:[10.1109/RE.2017.13](https://doi.org/10.1109/RE.2017.13).
- S. Türpe, A. Poller (2017) Managing security work in scrum: tensions and challenges. SecSE@ ESORICS 2017:34–49.
- Tøndel, I.A., Cruzes, D.S., Jaatun, M.G., 2020a. Achieving "Good Enough" software security: the role of objectivity. In: EASE '20: Proceedings of the Evaluation and Assessment in Software Engineering, pp. 360–365. doi:[10.1145/3383219.3383267](https://doi.org/10.1145/3383219.3383267).
- Tøndel, I.A., Cruzes, D.S., Jaatun, M.G., 2020b. Using Situational and Narrative Analysis for Investigating the Messiness of Software Security. In: ESEM '20: Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 1–6. doi:[10.1145/3382494.3422162](https://doi.org/10.1145/3382494.3422162).
- Tøndel, I.A., Jaatun, M.G., 2020. Towards a conceptual framework for security requirements work in agile software development. Int. J. Syst. Softw. Sec. Prot. 11 (1), 33–62. doi:[10.4018/IJSSSP.2020010103](https://doi.org/10.4018/IJSSSP.2020010103), IJSSSP.
- Tøndel, I.A., Jaatun, M.G., Cruzes, D.S., 2020c. IT security is from mars, software security is from venus. IEEE Secur. Priv. 18 (4), 48–54. doi:[10.1109/MSEC.2020.2969064](https://doi.org/10.1109/MSEC.2020.2969064).
- Tøndel, I.A., Jaatun, M.G., Cruzes, D.S., Moe, N.B., 2017. Risk centric activities in secure software development in public organisations. Int. J. Sec. Softw. Eng. 8, 1–30. doi:[10.4018/IJSSE.2017100101](https://doi.org/10.4018/IJSSE.2017100101), IJSSE4.
- van der Stock, A., Cuthbert, D., Manico, J., Grossman, J.C., Lang, E., (2021) OWASP application security verification standard 4.0.3, ed.
- van der Veer, R., (2019) SAMM agile guidance. <https://owasp.org/guidance/agile/>
- van der Heijden, A., Broasca, C., Serebrenik, A., 2018. An empirical perspective on security challenges in large-scale agile software development. In: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. Oulu, Finland. ACM doi:[10.1145/3239235.3267426](https://doi.org/10.1145/3239235.3267426).
- van Wyk, K.R., McGraw, G., 2005. Bridging the gap between software development and information security. IEEE Secur. Priv. 3, 75–79. doi:[10.1109/MSP.2005.118](https://doi.org/10.1109/MSP.2005.118).
- Venson, E., Alfayez, R., Gomes, M.M., Figueiredo, R.M., Boehm, B., 2019. The impact of software security practices on development effort: an initial survey. In: Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), IEEE doi:[10.1109/ESEM.2019.8870153](https://doi.org/10.1109/ESEM.2019.8870153).
- Viega, J., 2020. 20 years of software security. Computer 53, 75–78. doi:[10.1109/MC.2020.3013685](https://doi.org/10.1109/MC.2020.3013685), Long Beach Calif11.
- Weir, C., Becker, I., Noble, J., Blair, L., Sasse, M.A., Rashid, A., 2020a. Interventions for long-term software security: creating a lightweight program of assurance techniques for developers. Softw. Pract. Exp. 50 (3), 275–298. doi:[10.1002/spe.2774](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. J. Cybersec. 6 (1), doi:[10.1093/cybssec/tyaa007](https://doi.org/10.1093/cybssec/tyaa007).
- Williams, L., Meneely, A., Shipley, G., 2010. Protection poker: the new software security "Game". IEEE Secur. Priv. 14–20, doi:[10.1109/msp.2010.58](https://doi.org/10.1109/msp.2010.58), 8.3.
- Xiao, S., Witschey, J., Murphy-Hill, E., 2014. Social influences on secure development tool adoption: why security tools spread. In: Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing, pp. 1095–1106. doi:[10.1145/2531602.2531722](https://doi.org/10.1145/2531602.2531722).
- Xie, J., Lipford, H.R., Chu, B., 2011. Why do programmers make security errors? In: Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), IEEE doi:[10.1109/VLHCC.2011.6070393](https://doi.org/10.1109/VLHCC.2011.6070393).
- Yin, R.K., 2018. Case study Research and Applications. Sage, p. 6e.
- Inger Anne Tøndel** is a senior research scientist at SINTEF Digital, Trondheim, Norway, and a Ph.D. candidate at the at the Department of Computer Science, Norwegian University of Science and Technology (NTNU), Trondheim. Her research interests include software security, security requirements, information security risk management, and smart-grid cybersecurity. Tøndel received an M.Sc. in telematics from NTNU in 2004.
- Daniela Soares Cruzes** is a professor at the Department of Computer Science, NTNU, Trondheim, Norway. Her research interests are agile software development, software security, software-testing processes, empirical research methods, theory development, and synthesis of software-engineering studies. Cruzes received her Dr.ing. in electrical and computer engineering with emphasis in empirical software engineering at the University of Campinas, Brazil, in 2007. She has two postdoctoral studies, one at the Fraunhofer center at the University of Maryland, College Park, and one at the Norwegian University of Science and Technology, Trondheim. She is a member of committees with various highly ranked international conferences and journals.
- Martin Gilje Jaatun** is a senior research scientist at SINTEF Digital, Trondheim, Norway, and an adjunct professor at the University of Stavanger, Norway. His research interests include software security, security in cloud computing, and security of critical information infrastructures. Jaatun received his Dr.Philos. in critical information infrastructure security from the University of Stavanger in 2015. He is vice chair of the Cloud Computing Association, vice chair of the IEEE CS Technical Committee on Cloud Computing, an IEEE Cybersecurity Ambassador, an IEEE Computer Society Distinguished Visitor, and a Senior Member of the IEEE.
- Guttorm Sindre** is a professor at the Department of Computer Science, NTNU, Trondheim, Norway. His-research interests are in requirements engineering, security requirements, and IT education and didactics. Sindre obtained his Ph.D. from the Norwegian Institute of Technology in 1990. He was the leader of Excited Centre of Excellent IT Education from 2016 to 2021.

**G**

## **Paper H: ‘Continuous software security through security prioritisation meetings’**

Included is the published material [33], following the Creative Commons Attribution 4.0 International (CC BY 4.0) licensing arrangement used by Elsevier. The online supplementary material in Appendix B is included after the article.

H



Contents lists available at ScienceDirect

## The Journal of Systems &amp; Software

journal homepage: [www.elsevier.com/locate/jss](http://www.elsevier.com/locate/jss)

## In practice

Continuous software security through security prioritisation meetings<sup>☆</sup>Inger Anne Tøndel<sup>\*</sup>, Daniela Soares Cruzes

Norwegian University of Science and Technology (NTNU), Department of Computer Science, Sem Sælandsvei 9, Gløshaugen, 7034 Trondheim, Norway

## ARTICLE INFO

## Article history:

Received 22 March 2022

Received in revised form 18 June 2022

Accepted 2 August 2022

Available online 18 August 2022

## Keywords:

Software security  
Security meeting  
Security prioritisation  
Security requirements  
Agile software development

## ABSTRACT

Software security needs to be a continuous endeavour in current software development practices. Frequent software updates, paired with an ongoing flow of security breaches, requires software companies to address software security throughout development and post deployment. Prescriptive software security approaches do not match well with agile software development and its emphasis on self-management. Agile approaches are however in favour of meetings as a coordination and problem-solving strategy. This article investigates the role of regular security meetings centred on making security priorities and decisions for achieving continuous software security. Through technical action research and an observational case study, we studied variations of such meetings in three companies. We found that such meetings can reach key stakeholders, make security more visible, and contribute to ongoing security prioritisation. Thus, security meetings are a promising approach, especially for small and medium sized development companies with basic yet immature security competence. Future research should investigate further the role of such meetings and how best to organise them for different contexts and needs. For this we outline implications for research and practice, e.g., related to participants and how to organise the discussions and prioritisations in the meeting.

© 2022 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Contemporary software development happens as a continuous flow of software development rather than as larger planned increments. Further, software products are updated throughout their whole lifetime, to meet customer demands for new and improved features and to ensure continuous quality. Pairing this with daily news of security breaches, it becomes clear that software security needs to be a continuous endeavour as well.

As Fitzgerald and Stol (2014), we use *continuous* to represent a “holistic endeavor” and the “entire software life-cycle”. They define continuous security as, “Transforming security from being treated as just another non-functional requirement to a key concern throughout all phases of the development lifecycle and even post deployment” (Fitzgerald and Stol, 2014).

Continuous security does not come without effort. For security experts it would thus be tempting to prescribe security activities and tools to use during development and beyond, to ensure security is properly addressed. Research, however, shows that such prescriptive approaches are challenging to pair with the self-management of agile software development (ASD) (Turpe and

Poller 2017; (Weir et al., 2020a), and rather suggest “sensitizing the developers to their security needs, allowing them to choose for themselves which tools and techniques to use” (Weir et al., 2020a).

There are several frameworks and maturity models available for software companies wanting to continuously address software security, here exemplified by the OWASP Software Assurance Maturity Model (SAMM) (Crawley et al., 2020) and the Building Security In Maturity Model (BSIMM) (Migues et al., 2021). Both are agnostic to the development approach, and thus are relevant also for ASD (van der Veer, 2019). However, the comprehensiveness of these models (e.g., BSIMM12 now consists of 122 activities within the domains of governance, intelligence, SSDL touchpoints, and deployment) can make them hard to approach, especially for smaller companies that do not necessarily have the resources to build a large security program (Tøndel et al., 2020). And research points to small and medium-sized enterprises (SMEs) as having the largest potential for improvement of software security (Weir et al., 2020a). Further, knowing which activities to apply is not straight-forward. Being too ambitious may lead to an overspending on security (Tøndel et al., 2020), which can have negative business implications. For companies, what is considered adequate and cost-effective when it comes to security may vary between projects and change over time (McGraw et al., 2013; Tøndel et al., 2020), as development progresses and requirements

<sup>☆</sup> Editor: Dr. Daniel Mendez.

<sup>\*</sup> Corresponding author.

E-mail addresses: [inger.anne.tondel@ntnu.no](mailto:inger.anne.tondel@ntnu.no) (I.A. Tøndel), [daniela.s.cruzes@ntnu.no](mailto:daniela.s.cruzes@ntnu.no) (D.S. Cruzes).

<https://doi.org/10.1016/j.jss.2022.111477>

0164-1212/© 2022 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).



are negotiated. Thus, there is a need for strategic decisions on security on a regular basis.

ASD is oriented towards people, interactions, and self-management (Beck et al., 2001), with meetings as a major mean of coordination (Strode et al., 2012). To exemplify, Scrum (Schwaber, 2004) is largely centred on meetings and include five meeting types: sprint planning meeting, daily Scrum meeting, sprint review meeting, sprint retrospective meeting, and product backlog refinement. None of these meetings are focused on security. Some researchers have proposed regular meetings on security in ASD, in form of adding security review meetings to Scrum when necessary (Kongsli, 2006), using Protection Poker to collectively estimate the security risk in every iteration (Williams et al., 2010), or organising a Security Intention Meeting Series to regularly involve decision makers in security prioritisation (Tøndel et al., 2019a). Further, less regular meetings are proposed in form of Threat Modelling sessions (Bernsmed et al., 2022) or security workshops aimed toward security incentivisation (Weir et al., 2020a). Many of the activities included in BSIMM or OWASP SAMM could be performed through or supported by regular meetings, examples being Threat Assessment, Security Architecture, and Architecture Assessment in OWASP SAMM. However, existing research points to uncertainties in how to best organise such meetings to ensure effect on the software security (Cruzes et al., 2018; Tøndel et al., 2019b). A better understanding is needed on what are the effects of security meetings related to development, and how practitioners can be guided in ensuring effect from meetings. Furthermore, previous studies have identified challenges regarding longer-term adoption of security meetings (Tøndel et al., 2019b; Weir et al., 2020a, 2021; Bernsmed et al., 2022), leading to the need for more knowledge of how to support ongoing adoption.

This article proposes and studies regular security meetings that: (1) are not confined to security experts but rather include key decision makers as participants, (2) identifies and assesses security needs, and makes prioritisations and decisions on the next steps, and (3) are flexible and can be adjusted to the needs and priorities of the company when it comes to meeting scheduling and organisation. As such, we build on the previous suggestions for a Security Intention Meeting Series (Tøndel et al., 2019a). The research we report on is part of a design science study aimed at improving software security prioritisation by developing a meeting approach that satisfies the needs of ASD projects. We study such security meetings in three SMEs, one of which were already running this type of meeting, and two where we brought this meeting type to the company. Our main research question is the following: *How can regular security meetings centred on making security prioritisations and decisions be organised to maximise their positive effect on the priority given to security?* (RQ1).

We have previously described the concept *security prioritisation* as “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, we take a broad view of security prioritisation. To relate to this rather intangible concept of security prioritisation, it is necessary to concretise what security prioritisation may look like in a project, and what can be done to influence the priority given to security. As part of this design science study, we have previously performed a case study to investigate what influences the security prioritisation throughout an ASD project. We found that the priority given to security was influenced by the presence of a driving force for security, the visibility of security, the motivation, the room to manoeuvre, and the process match (Tøndel et al., 2022). In the study reported in this paper, we use these previously identified influence categories to support us in understanding how the studied meetings can have a positive

effect on the priority given to security. Additionally, we study *what effects are seen by adopting this type of meeting* (RQ2) and *what facilitates or hinders the adoption of such meetings* (RQ3).

The studied meeting instances varied in their structure, the support offered, and in who participated. The companies varied in size, development approach, and in customer relations. This variety allowed us to identify similarities and variations across the cases, and use this to understand: (1) what are common experiences that a broader set of companies might expect from applying this meeting concept, and (2) how key variations can be explained based on the cases. The article contributes both to practice and research: (1) we use the lessons learned to provide development companies with better support in deciding whether to take up regular security meetings in development, and how to organise such meetings, and (2) we provide researchers with insight into the practical experiences in performing such meetings and point to research needs.

The article is organised as follows. Section 2 uses literature to motivate regular software security meetings, as well as presents current knowledge on security meetings in ASD. Section 3 describes the research approach, including the cases studied. Section 4 presents the findings according to the three research questions. Section 5 discusses the implications of these findings, Section 6 discusses the threats to validity, and Section 7 concludes the article.

## 2. Background and related work

This section uses current literature on software security and on meetings in ASD to explain the theoretical background for investigating regular security meetings. Furthermore, it describes the known evidence from studies of similar types of meetings, and introduces in more detail the Security Intention Meeting Series that we build on in this work.

### 2.1. Background for investigating regular security meetings

There is a growing body of literature on how to integrate security and other software qualities with ASD. This includes literature on working with security requirements in ASD (Villamizar et al., 2018; Tøndel and Jaatun, 2020), managing quality requirement sin ASD (Behutiye et al., 2020); (Jarzębowicz and Weichbroth, 2021), and on bringing ASD to safety-critical systems (Heeager and Nielsen, 2018). Literature reviews within this area point to some recurring challenges of working with security and other quality requirements in ASD. One challenge identified by many studies is that of neglect of quality requirements (Behutiye et al., 2020); (Jarzębowicz and Weichbroth, 2021). Related challenges are that of late consideration of quality requirements (Behutiye et al., 2020) and a lack of recognition by stakeholders (Jarzębowicz and Weichbroth, 2021). Proposed solutions include initiatives to start focus on quality requirements earlier in the project, and to involve multiple roles and viewpoints in eliciting and reviewing quality requirements (Jarzębowicz and Weichbroth, 2021). Still, a main criticism towards much of the existing research in this field is the limited empirical evaluations of proposed techniques and approaches to integrate software security into ASD (Villamizar et al., 2018; Bishop and Rowland, 2019; Behutiye et al., 2020). There is a call for more guidelines, not only practices and methods (Behutiye et al., 2020). Furthermore, there is still a need for more lightweight strategies, as the challenge of time constraints due to short iterations have not received adequate attention in the proposed strategies, despite this being a commonly reported challenge (Behutiye et al., 2020).

Literature provides some knowledge on what can increase the priority given to security in an agile development context. Newton et al. (2019) studied literature and performed interviews to

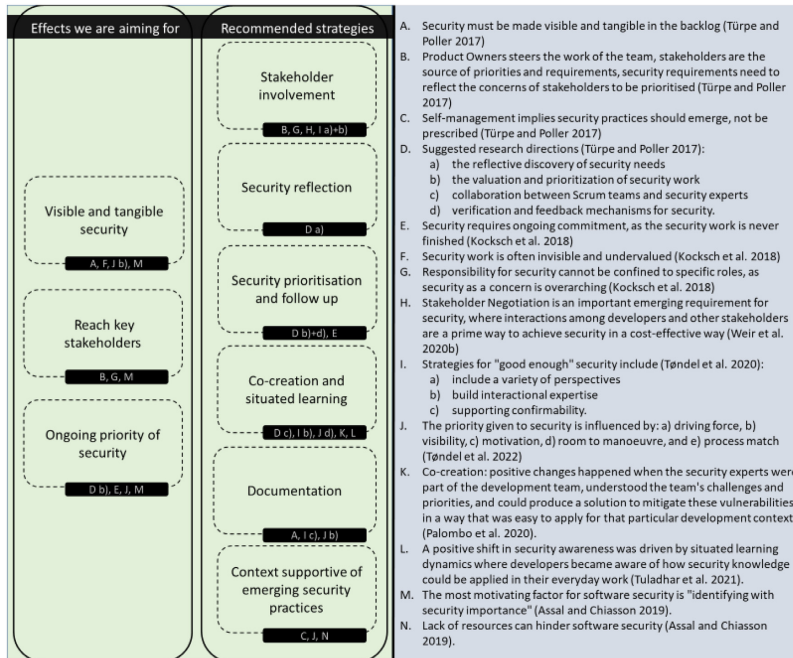


Fig. 1. Literature supporting the suitability of regular security meetings to achieve prioritisations and decisions regarding security.

identify success factors for software security in ASD. They pointed out the importance of practices centred around individuals and culture. This is in line findings in the previously mentioned case study (Tøndel et al., 2022), where we identified five areas that influence software security prioritisation in an ASD project. Many of these influence areas (driving force, visibility, motivation, room to manoeuvre, and process match) are related to individuals and culture.

Türpe and Poller (2017) explored tensions between Scrum and security requirements through a case study. They identified challenges related to key stakeholders such as product owners. They pointed out that it is important to make security visible as a concern, and they emphasised the need for more research on prioritisation of security and on improving collaboration between agile teams and security experts. Drawing on data from ethnographic studies, Kocksch et al. (2018) explained how security's similarities with care work (invisible, undervalued, never finished) can make recognition of security work challenging. Furthermore, they pointed out how security needs to be handled in collaboration among a broad set of stakeholders. Building on theory on objectivity, Tøndel et al. (2020) identified the inclusion of a variety of perspectives and the building of interactional expertise as key strategies to move towards "good enough" security.

Assal and Chiasson (2019) performed a survey aimed at understanding how to support developers in their work on software security. They found that developers need to identify with the importance of security. In this respect, the ethnographic studies of Palombo et al. (2020) and Tuladhar et al. (2021) pointed to the role of co-creation and situational learning (respectively) in changing software security practices. It is not enough to point

developers to the presence of security problems. Rather, changes are seen when developers and security experts work together to solve problems. This is in line with Weir et al. (2020b), who identified the need to move towards a "dialectic" approach to security. Building on data from interviews they identified interactions among developers and other stakeholders as a prime way to achieve security in a cost-effective way.

Fig. 1 provides our summary and synthesis of the findings from the above-mentioned studies. On the right side of this figure, we list key findings from the studies. Then, on the left side of the figure, we synthesise these findings into a set of effects called for in literature, and a set of recommended strategies. Letters are used to link the findings from the studies (on the right) with the effects/strategies (on the left). As shown in the figure, there is a call for more visible and tangible security (Türpe and Poller, 2017; Kocksch et al., 2018; Assal and Chiasson, 2019; Tøndel et al., 2022), for reaching key stakeholders with security (Türpe and Poller, 2017; Kocksch et al., 2018; Assal and Chiasson, 2019), and for making security an ongoing priority (Türpe and Poller, 2017; Kocksch et al., 2018; Assal and Chiasson, 2019; Tøndel et al., 2022). Recommended strategies for security include stakeholder involvement (Türpe and Poller, 2017; Kocksch et al., 2018; Tøndel et al., 2020; Weir et al., 2020b), security reflection (Türpe and Poller, 2017), security prioritisation and follow-up (Türpe and Poller, 2017; Kocksch et al., 2018), co-creation and situated learning (Türpe and Poller, 2017; Palombo et al., 2020; Tøndel et al., 2020; Tuladhar et al., 2021; Tøndel et al., 2022), documentation for security (Türpe and Poller, 2017; Tøndel et al., 2020, 2022), and having a context supportive of emerging security practices



(Türpe and Poller, 2017; Assal and Chiasson, 2019; Tøndel et al., 2022)

Security meetings are not a goal, but a possible mean to achieve ongoing and strategic prioritisation of security. Regular security meetings are likely to be able to support the effects and strategies identified in Fig. 1, e.g., by involving stakeholders on security and engaging them in regular security reflection, prioritisation, and follow-up. Furthermore, addressing security through meetings is highly compatible with an agile approach. According to the Agile manifesto, “The most efficient and effective method of conveying information to and within a development team is face-to-face conversation” (Beck et al., 2001). As already mentioned, Scrum relies on several meetings (Schwaber, 2004). Daily stand-up meetings are also recommended within Kanban (Ahmad et al., 2013). In ASD, meetings support teamwork quality through balancing member contributions and facilitating mutual support (Lindsjörn et al., 2016), and offers a major mean of coordination (Strode et al., 2012; Moe et al., 2018).

## 2.2. Related work on regular security meetings in ASD

Despite meetings playing a central role in ASD, there is limited research on the effect and organisation of meetings that involve activities related to ongoing prioritisation, planning, and follow-up. Of such meetings specific to ASD, the daily stand-up meeting has been most extensively studied by Stray et al. (2016).

When it comes to security meetings directed towards agile or continuous software development, we have identified the following approaches:

- Security Review (SR) meeting (Kongsli, 2006): Arranged after the Scrum iteration planning meetings in cases where there were many or complex security concerns related to the user stories that were picked. The full development team participated in this meeting to ensure collective ownership also of security issues.
- Protection Poker (PP) (Williams et al., 2009, 2010; Tøndel et al., 2019b): A collaborative risk estimation game that gathers the whole development team to discuss, identify and rank the software security risks related to the features to be implemented in the upcoming iteration.
- Threat Modelling (TM) meetings, as studied by Cruzes et al. (2018) and Bernsmed et al. (2022): Meetings centred on performing threat modelling, e.g., using Data Flow Diagrams and the STRIDE mnemonic.
- Facilitated Security Workshops (SW) (Weir et al., 2020a, 2021): Workshops centred on incentivisation, threat assessment, and on-the-job training. Used the Agile Security Game, a simplified threat assessment, and follow-up sessions, to discuss security issues and questions. Workshops were led by researchers who were not security experts, to study the effect of this workshop package also when there were no security experts available.
- The Security Intention Meeting Series (Tøndel et al., 2019a): A meeting series to gather key decision makers in a project, to regularly assess the state of the software security of the project and identify concrete actions moving forward.

Fig. 2 gives an overview of identified effects from these meeting types, as well as what has been found to work well or be challenging. The figure is organised so that findings from studies of the security meetings are presented together, while findings on the daily stand-up meetings are presented separately. In the following, we first present aspects related to meeting organisation, before moving on to output and effect, and, finally, point to facets of the context.

When it comes to meeting organisation, many of the aspects of the meetings that worked well were related to strategies called for in existing literature (see Fig. 1). Examples are stakeholder involvement (participation by the full team Kongsli, 2006; Williams et al., 2010; Weir et al., 2020a; Bernsmed et al., 2022, facilitation by managers Weir et al., 2021), security reflection (discussions and active participation Tøndel et al., 2019b; Bernsmed et al., 2022), and co-creation and situated learning (peer-based learning Weir et al., 2021). This points to meetings as a powerful intervention, which should be properly addressed in research. Broad participation and discussions were pointed out as important across the different security meetings. Broad participation however came with the risk of less effective meetings (Stray et al., 2016). Both Protection Poker and Threat Modelling found clear needs for some security expertise, e.g., to be able to explain terms and ensure quality (Cruzes et al., 2018; Tøndel et al., 2019b; Bernsmed et al., 2022). This contrasts with the Security Workshops, which had as a requirement that they should work also without security experts (Weir et al., 2020a). The challenges and uncertainties identified related to meeting organisation (e.g., uncertainties on how to structure the meeting and who to include in order to make the meetings effective Cruzes et al., 2018; Tøndel et al., 2019b) points to a need to further explore different meeting types and collect more experiences to guide both researchers and practitioners. This article meets this need.

When it comes to effect and output, both the daily stand-up meeting and the security meetings could help get overview of issues and solve problems/make improvements. Related to the desired effects outlined in Fig. 1, the meetings generally contributed to making security more visible and tangible. Despite the meetings leading to security improvements in processes and code, studies point to challenges in following up the risks identified in the meetings and seeing how the meeting output is linked to improved security of the products (Cruzes et al., 2018; Tøndel et al., 2019b). There is a need to understand better what makes this transition challenging, so that better guidance can be offered. Existing literature (see Fig. 1) points to following up of prioritisations, something that can be done in meetings. However, there are likely more complex reasons that make this transition challenging. To exemplify, both Palombo et al. (2020) and Weir et al. (2020a) emphasise the systemic aspect of software security. The study reported on in this article examines how effects from meetings can be supported or hindered (RQ1), considering aspects of the meeting as well as the context.

When it comes to the context, the studies of these meetings pointed to the importance of motivation and time for security (Tøndel et al., 2019b; Weir et al., 2020a; Bernsmed et al., 2022). Further, the company size and the security maturity level might be important for longer-term adoption. Weir et al. (2021), who studied eight organisations of varying size, found that adoption was strongest in the medium-sized organisations, followed by the smaller organisations, while adoption was low in the larger organisations. However, we have reason to believe that a broader set of contextual factors have implications for adoption of meetings and their effect. We base this expectation on the large number of documented challenges to software security and other quality aspects in ASD (Oueslati et al., 2015; Behutiye et al., 2020; Jarzębowski and Weichbroth, 2021), as well as the substantial amount of organisational blockers and motivators (Weir et al., 2020a) and influences (Tøndel et al., 2022) identified for adopting and prioritising software security practices. In this article we contribute with more knowledge on contextual factors important for getting effect and adoption of regular security meetings in development companies or teams. Further, as time has been identified as one key obstacle, we study meeting approaches where the schedule can be adapted to the time pressure experienced in the company.

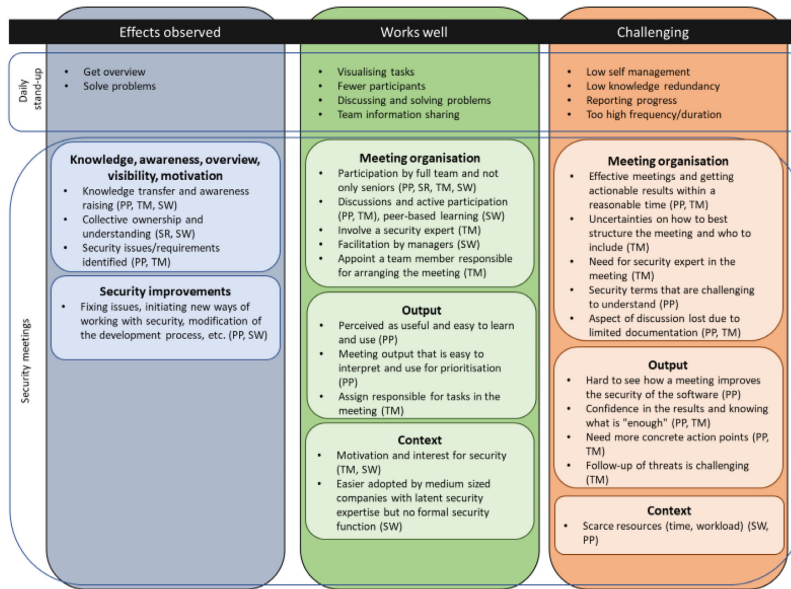


Fig. 2. Key findings from studies of the daily stand-up meeting, as well as the Security Review Meeting (SR), Protection Poker (PP), Threat Modelling (TM), and facilitated security workshops (SW).

2.3. The security intention meeting series

In this study, we decided to examine meetings that followed the spirit set out for the Security Intention Meeting Series (Tøndel et al., 2019a) but with freedom for companies to adapt the approach to match their needs. This is based on recommendations that developers should not be prescribed a particular way of addressing security, but rather be empowered to make their own decisions (Türpe and Poller, 2017; Weir et al., 2020a). The Security Intention Meeting Series approach was a response to challenges with getting companies to consider security in every iteration, e.g., as is done in Protection Poker, while needing a more lightweight and recurring approach than what is common for threat modelling and security risk analysis (Tøndel et al., 2019a). Further, this meeting approach had not yet been studied empirically.

The Security Intention Meeting Series approach (Tøndel et al., 2019a) can be summarised as follows. Early in the project, key decision makers are gathered, together with people knowledgeable about security and development, to discuss the security intentions of the project. This implies agreeing on the security goals and needs of the project, and what aspects need to be given particular attention (*intention setting*). Then, regular follow-up meetings are arranged throughout. These consists of a *status assessment* ("Are we moving towards our goals regarding software security?") and an identification of *action points* for the next period ("What will be our concrete priorities moving forward?"). Companies and projects are, however, free to adapt the meeting approach to their needs, e.g., regarding how often to arrange such meetings, who should facilitate the meetings, and who should be invited as participants. Still, some guidance is given. Shorter meetings are preferred to longer ones, there should be some regularity to the meetings (e.g., by always agreeing on a time for

the next meeting as part of the meeting), one person should be responsible, and roles such as product owner or project manager should be present.

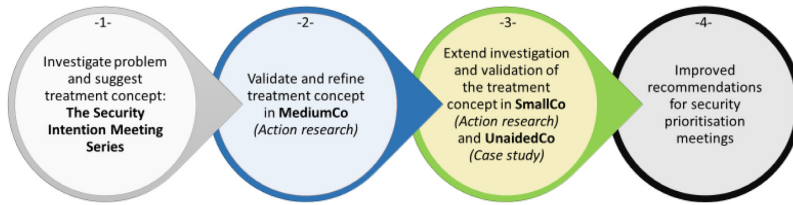
3. Research approach

First, this section describes the overall research approach of this work as that of design science, and explains how technical action research and case studies were used in combination to support the design goal. Then, it introduces the research design choices that were made for both the technical action research and case study research parts of the study, before it describes each of these parts in more detail. Finally, the analysis approach is presented.

3.1. Combining technical action research and observational case studies into a design science research approach

The main research question of this work represents a design goal, aimed at improving software security prioritisation by developing a meeting approach that satisfies the needs of agile development projects. Consequently, the overall research approach of this work is design science (Wieringa, 2014). Design science "is the design and investigation of artifacts in context", where the artefacts "interact with a problem context in order to improve something in that context" (Wieringa, 2014). Thus, design science iterates between two problem-solving activities: (1) designing artifacts to bring about improvements, and (2) answering knowledge questions about the context and the artifact in the context. To answer the knowledge questions, the researcher can bring in other research methods (Wieringa, 2014). Two of the possibilities are observational case studies ("a study of a real-world case without performing an intervention" (Wieringa, 2014) and technical





**Fig. 3.** The overall research approach, using design science with technical action research and observational case studies to validate and refine the recommendations for security prioritisation meetings.

action research (“the use of an experimental artifact to help a client and to learn about its effects in practice” (Wieringa, 2014)). Both observational case studies and technical action research can provide understanding of the underlying mechanisms that produce real-world phenomena (Wieringa, 2014). Thus, both these research approaches were suited to understand the adoption and the effects of these meetings (the artefact) and what brought about this adoption and effect.

The overall design science approach is depicted in Fig. 3. The concern of this article is step 2–4 in this figure. The decision to combine action research and case study research was pragmatic; we used technical action research with the companies we studied that did not already perform regular security prioritisation meetings and used observational case study with a company that was already performing such meetings. When action research was applied, our goal in the study was both to help the company with their security and to validate and refine the meeting concept, and we iterated upon and adapted the meeting approach throughout the study to better meet the needs of the company. When case study research was applied, we studied a meeting approach already in use by the company without aiming to improve upon the approach or help the company in their approach to software security.

### 3.2. Overarching research design choices

The Security Intention Meeting Series approach (Tøndel et al., 2019a) that we wanted to validate and refine through this study had already been designed. However, as our goal was to improve this approach, we were free to adapt this meeting concept to the needs of the companies we interacted with. We see this as a strength of our study. There is no one single way to achieve security prioritisation through regular security meetings. The size of the company and the type of project and customer might impact both how to run the meeting effectively, what kind of support would be needed, e.g., in form of a template, and who should participate. Thus, the meetings we studied all shared key characteristics with the original idea of the Security Intention Meeting. Still, we allowed for variation from these characteristics on some aspects depending on the needs of the company. Thus, there are some discrepancies in who participated in the meetings and in the setting of intentions, compared to the original design of the approach. An overview of the characteristics of the studied meeting models is given in Table 1.

To perform this study, we needed to recruit companies with: (1) an intention to perform ongoing security prioritisation, and (2) an ongoing security prioritisation meeting initiative or a willingness to initiate such a meeting initiative. Through a research project with several company participants, we had access to several cases that matched these needs, and we opted to involve three companies in the study. The companies have been given the pseudonyms MediumCo, SmallCo, and UnaidedCo for the purpose

of this article. An overview of characteristics of these companies is given in Table 2. This variety of companies allowed us to evaluate this meeting type in companies ranging from very small to medium size, and with different customer relations that in varying ways constrained their ability to incorporate regular security meetings as part of development. Further, it allowed us to study how to start applying such a meeting, as well as study a meeting that had already been successfully adopted by a development company. The choice to focus on SMEs was guided by literature showing smaller companies have a larger potential for improvements in their software security than larger companies, and showing more success with security meetings in smaller companies (Weir et al., 2020a).

For all cases, we used multiple data collection methods, as is recommended for case studies (Yin, 2018). An overview is given in Table 3. The action research study was centred on meetings that we facilitated and observed. Meeting observations were supplemented with other data sources, including interviews. For the observational case study, we used a similar approach where observation of meetings was a central part of data collection. All data collection was done by the first author, and in the cases where we used action research, the second author had the role of facilitator of the meetings. In MediumCo and UnaidedCo the observer was largely passive in meetings, while the observer participated more to the discussion in SmallCo. The first author took detailed notes from all security meetings, including notes on the structure of the meetings, the topics that were discussed, the types of security decisions and priorities made, the participants' level of engagement, what worked well, what was challenging, and if anything was surprising. We also reflected on the potential influence of the observer in the meeting. Interviews were semi-structured, and covered topics such as the goal of the security meetings, their effect, how the meetings could be improved, and the intention to continue with the meetings. With UnaidedCo, the retrospective was led by the first author and covered similar topics as the interviews. More information on the data collection instruments is given in Appendix B.

The main ethical aspects of this study are the privacy of the individuals participating in the study, the sensitive information on the security of the solutions as shared with us in meetings, and ensuring volunteer participation in the study. Privacy related to data collection and analysis was specified in a report sent to the Norwegian Centre for Research Data, an organisation that provides data protection services to Norwegian research institutions. This ensured that data handling plans were in accordance with current privacy legislation. Observation notes were made in such a way that individuals were not directly identified. Interviews were only recorded upon interviewees informed consent. When it comes to security of sensitive company data, this was ensured through non-disclosure agreements (NDAs) with the companies. In observations, we took care not to write down what the participants pointed out as highly sensitive. Access to the raw data and



**Table 1**  
Meeting model characteristics.

Meeting model characteristics	MediumCo	SmallCo	UnaidedCo
Meeting maturity	Initiated in one project, then continued in another project, and eventually brought to another team.	Not used before.	Had run this type of meeting for 10 months when we started observation.
Scope	Project/team	Project	Department
Participants from the company	<i>Initial use:</i> 3–5 people from the following roles: security resources (security officer, security champion), product owners (mainly those with more technical background), developer representatives. <i>Brought to new team:</i> the product owner and the full team.	Developers (1–2)	Department lead and system architects (5)
Physical/online facilities	<i>Initial use:</i> mixture of physical and online participants, shared screen <i>Brought to new team:</i> online meeting with shared screen	Online meeting with shared screen	Physical meeting, one location, screen shown in meeting
Facilitation	<i>Initial use:</i> security officer <i>Brought to new team:</i> product owner	External security expert	Department lead
Frequency	<i>Initial use:</i> Monthly – time for next meeting decided upon in the meeting <i>Brought to new team:</i> NA	Biweekly – time for next meeting decided upon in the meeting	Monthly
Duration	45 min–1 h	Initially 1 h. Later 30 min.	2 h scheduled, usually spent 1 h and 30–45 min
Support material	Confluence page with security areas and supporting questions.	Excel sheet with security areas and supporting questions.	NA
Meeting documentation	Confluence page: concerns and action points added within the structure of the support material	Excel sheet: concerns and action points added within the structure of the support material	Meeting memos with action points + excel sheet with overview of all identified security concerns that were not yet fully addressed
Typical agenda	(1) Status of tasks and open issues from previous meetings; (2) Discuss security areas not previously addressed, or where there are open issues still; (3) Time for next meeting.		(1) Status of action points from last meeting; (2) Open discussion on security issues; (3) Excel sheet with security concerns; (4) New action points.

**Table 2**  
Company characteristics.

Company characteristics	MediumCo	SmallCo	UnaidedCo
# developers	About 80 developers	2–3 developers	About 20 developers
# locations with developers	4	1	1
Criticality of security	( <i>Medium</i> ) Clear security risks, mainly related to offering a public service to many users, and in ensuring validity of tickets.	( <i>Low</i> ) Limited security risks but with plans to develop new solutions that brings in both privacy and security concerns.	( <i>High</i> ) Develops solutions that handle security critical information.
Customer relation	Targets mainly one sector. Bid process in competition with other actors. Varying security concerns among customers.	Several smaller customers without much security competence and concerns. New bigger customer upcoming.	One main customer (the mother company) that is concerned about security.
Agile principles adherence	Hybrid. Scrum-based development process but with rather fixed contracts.	Few developers, thus few clear processes.	Hybrid. Kanban-based processes within the development department.
Presence of central security support	Initially: security officer as part of the development department and security champions in development teams. Later: key resources left without being replaced.	NA	Chief Security Officer in mother company (the customer). One person in the development department with an informal security role.



**Table 3**  
Overview of data collection.

Data collection	MediumCo	SmallCo	UnaidedCo
Main research approach	Action research	Action research	Case study
Observations of security meetings (documented in an observation template)	7 (one of which were shortened/largely skipped) (11/2019–08/2020)	11 meetings (10/2020–03/2021)	4 (09/2019–03/2020)
Other observations (documented in an observation template)	2 meetings to bring this technique to new team	3 introductory meetings before starting with the meeting template: 1 visit to the company	NA
Interviews (interviews at MediumCo and SmallCo were recorded and transcribed upon interviewee consent; interviews and retrospective at UnaidedCo were performed by two researchers, where one was responsible for taking notes)	Interviews with two product owners after participating in meetings (online) (06/2020)	Interview with the main developer after participating in meetings (online) (05/2021)	Interview with 2 meeting participants and 3 outsiders before observations started (in-person) (04/2019); retrospective after observations (2 sessions, 4 participants from the meeting + one outsider, in-person but with one online participant) (09/2020)
Status updates (documented in notes or in emails)	6 informal talks with the security officer (08/2019–06/2020); email exchange with one product owner (08/2020)	Email exchange with main developer on adoption of the technique 10 months after the other data collection (01/2022)	NA
Documentation	Example confluence page; description of security areas and security questions	Excel sheet used as template for meetings; description of the meeting approach written by the main developer; some of the security material developed as a result of the meeting	NA

the analysed data was limited to a few individuals. Participation in the study was voluntary for the companies and the individuals. However, we recognise that for MediumCo and SmallCo, participation in the study led to them getting support in their software security work. This may have made it more difficult for them to refuse participation. For all companies, it might also be challenging for individuals to refuse participation if the company was part of the study. However, we got the impression that participants were positive towards contributing to this research.

### 3.3. Technical action research: MediumCo and SmallCo

Technical action research relies on mutual trust, and such trust can take a long time to establish (Wieringa, 2014). When we recruited the first company, MediumCo, the meeting concept had not yet been tried out in a company and we thus needed to work with a company where we had predefined trust to try out new ways of working. MediumCo matched this need. Furthermore, it was a good case as it had characteristics that we suspected to be common among SME; it had few dedicated security resources, an ongoing yet immature software security initiative, cross-border working arrangements, and operated in a strongly competitive business. Thus, it was a relevant case to study, also if we would end up with a single-case design (Yin, 2018). We had worked with MediumCo before and had good knowledge of the company and its context, something that reduced one of the common challenges in canonical action research, i.e., to deal with the organisational complexity when diagnosing the current situation (Davison et al., 2012). This also made it easier for us to make an initial instantiation of the meeting concept that matched this company.

MediumCo and its development was organised in several teams, and projects could be performed by one or more teams. Each of these teams had one or two product owners associated with the team. A security officer role was part of the development department, overseeing and supporting the security work in the teams. In addition, each team had one assigned security champion

– a developer with extra attention to security issues. We started working with a team (in the following referred to as Team A) where the product owners were technically skilled and interested in security, thus making them open to try this technique. The security officer was active in adapting the security intention meeting concept to the needs of MediumCo.

In the instantiation of the security intention meeting for MediumCo, the status assessment was supported by a checklist consisting of some general questions and a long list of security areas to consider (See Appendix A). This way the meeting participants were supported in identifying all the important security issues to be considered. However, this made the intention setting be more technical and thorough than originally envisioned (Tøndel et al., 2019a). In the meetings, participants assessed the status of already identified security tasks, discussed security needs and progress related to the security areas in the checklist, and identified priorities moving forward, including a time for the next meeting. Participants were product owners, the security officer, and key representatives from development.

As illustrated in Fig. 4, we facilitated and observed five meetings in Team A. Then we performed interviews with the two product owners that were considered the key participants in these meetings. Based on the experiences from Team A, the company wanted to bring the meeting to another team (Team B). An introduction to the meeting approach was given by one of the product owners involved in Team A. In this case, no researcher was involved as facilitator. Instead, the Product Owner of Team B did the facilitation. The meeting concept was the same, however, in this case participants included the full team.

Experiences from MediumCo made us interested to see how this meeting approach would work in a company with less resources dedicated to security. This led to the recruitment of SmallCo, a very small development company with close to no previous experience in software security. They were motivated to participate, as envisioned changes to their software product portfolio made it necessary to incorporate more security into their development activities. Thus, working with SmallCo represented

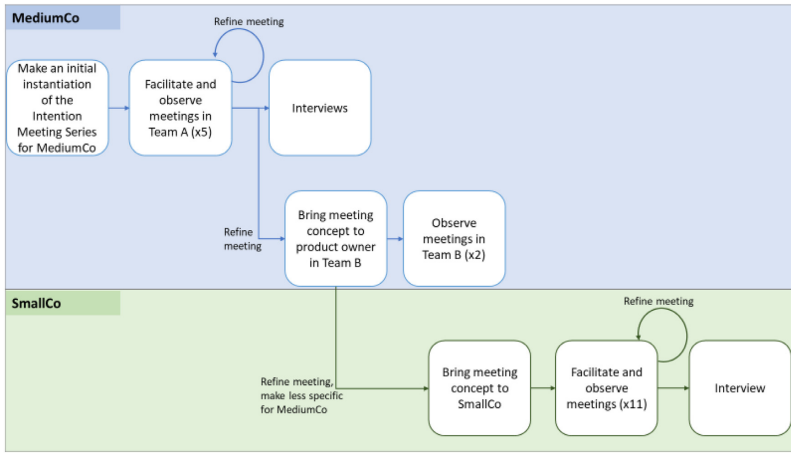


Fig. 4. Overview of the technical action research with MediumCo and SmallCo.

an opportunity to see how regular security prioritisation meetings could support companies with limited security resources and experiences, a situation we envision to be common in many SMEs.

After having iterated over the meeting approach with MediumCo, the meeting support material used was refined to make it less company specific. Then it was brought to SmallCo. Initially, we spent some time getting to know the company, as we had not worked with this company before. This took the form of meetings and a company visit. Then, we facilitated eleven meetings, followed by an interview with the main developer of SmallCo. The meetings had a similar structure as in MediumCo, but with only developers as participants. However, due to the limited size of this company these developers had roles also related to security, and they were involved in strategic discussions with managers in the company.

The number of meetings facilitated and observed was based on practical considerations as well as principles related to saturation. In MediumCo, we facilitated enough meetings in Team A to make the company confident that they could continue with meetings if they chose to. When meetings were brought to Team B, we observed all meetings that were performed. In SmallCo, we ended up achieving saturation in our observations, with no main new issues emerging in the last few meetings. More details on the meeting model and the support material used in MediumCo and SmallCo can be found in Appendix A and in Table 1.

Technical action research is different from other action research as it is artifact-driven, not problem driven (Wieringa, 2014). Still, it satisfies the principles of canonical action research (Davison et al., 2004; Wieringa and Morali, 2012). Table 4 provides an overview of how the technical action research, as applied in this study, relates to these principles.

3.4. Observational case study: UnaidedCo

Case studies study phenomena in their real-world context (Runeson and Höst, 2009; Yin, 2018), and in this case required a company already performing some sort of regular security prioritisation meeting. Through our interaction with companies, we identified a company that had such a practice. This happened as part of work we were doing with this company to identify and

evaluate their software security practices. At that point we were already doing technical action research with MediumCo, and we saw the opportunity to complement the knowledge gained from technical action research with a case study of a security meeting approach that were ongoing and led by the company itself.

The meeting approach had been developed by UnaidedCo independent of the ideas related to the security intention meeting series (Tøndel et al., 2019a). However, the meeting had many similarities with the original security intention meeting concept. Due to the organisation of the meeting at the department level, no product owners or similar were present (these were in a different department). Meeting participants consisted of the department manager as well as senior employees. Together this group had security competence, decision making authority, and knowledge of the development. There was less attention to the setting of intentions and following them up, as compared with the original idea. A typical meeting started with going through the status of previous action points. Then followed an open discussion on security concerns, with the aim to identify and note down any such concerns to inform security prioritisations. Then the participants decided on action points for the next period. The meeting happened monthly. More details on the meeting model and the support material used in MediumCo and SmallCo can be found in Appendix A and in Table 1.

Fig. 5 provides an overview of the case study with UnaidedCo. As already stated, it started with interviews aimed to identify and evaluate current software security practices. Then, we moved on to observing four security group meetings. These were facilitated by UnaidedCo, and the first author acted as an observer. Eventually, we arranged a retrospective that served as an opportunity for the company to discuss and improve upon their own meeting practice, as well as an opportunity to get feedback on initial findings from the observations. This way, we established a basic overview of the context in which the meetings took place, got deep knowledge on the meetings through observations, and got to know the participants' thoughts on the meetings and their effect.

The number of meetings observed (4) was agreed with the company beforehand. We however experienced that few new issues came up in the meeting observed last, indicating that we were moving towards saturation in the observations.



**Table 4**  
Adherence to the principles of canonical action research (Davison et al., 2004).

Canonical action research principle	Technical action research as performed in this study
1 – the principle of the researcher–client agreement	The action research was part of a bigger research project where we had established NDAs with the companies. The companies agreed that this meeting could be a good approach for them given their situation, and they agreed to participate in data collection.
2 – the principle of the cyclical process model (diagnosis – action planning – intervention – evaluation – reflection)	For both MediumCo and SmallCo, there were initial activities to understand their needs and assess the relevance of the security meeting approach as an intervention (diagnosis). Then, the meeting approach was instantiated for the company (action planning) before the meeting series started (intervention). In relation to the meeting there were reflections with the participants on the meeting itself, and the researchers also made their assessment of how the meetings could be improved (evaluation, reflection). Based on this, adjustments were made before the next meeting. Evaluation also happened in semi-structured interviews.
3 – the principle of theory	The action research was guided by the hypothesis that the artifact would be able to improve the security prioritisation in the companies. Previous knowledge on influences on security prioritisation were used to understand the effects of applying the artifact.
4 – the principle of change through action	Each cycle aimed to improve the security prioritisation of the company/project, as well as to improve the security meeting approach.
5 – the principle of learning through reflection	The researchers and the company representatives reflected on the meetings, both as part of the meetings themselves and in interviews.

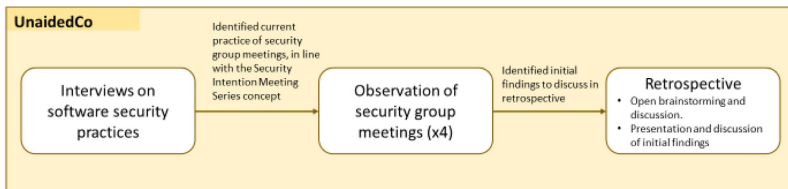


Fig. 5. Overview of the observational case study with UnaidedCo.

3.5. Analysis

The analysis approach we applied allowed us to dig deep into each case individually, while allowing for the necessary overview to identify findings and learning points across cases. The analysis process consisted of two main stages, as depicted in Fig. 6. The first stage was concerned with analysing each case individually. This was important, as a main strength of case studies – and we would claim, also of action research – is to be able to dig deep into cases and take the wholeness of the case into account (Yin, 2018). Thus, cross-case synthesis, the second analysis stage, should be performed with the goal “to retain the integrity of the entire case and then to compare or synthesise any within-case patterns across the cases” (Yin, 2018). These recommendations informed the analysis process of this study. We used the same analysis strategy for each case, including the same coding structure. However, synthesis was done on the aggregated findings from each case. These aggregated findings were documented in longer memos. This approach was chosen to ensure we did not perform a simplified comparison on the variable level but rather compared and synthesised findings on the case level (Yin, 2018). To support cross-case comparison, the findings from each case were summarised in a table in an excel sheet, as visualised in Fig. 6. This is in line with recommendations from Miles et al. (2018) of using matrix displays to support cross-case analysis. This table provided an overview of the findings related to the effect identified from the meetings (RQ2), what was found to increase or reduce this effect (RQ1), and what contributed to or hindered adoption (RQ3). For each entry in the table, it was stated which case (marked M (MediumCo), S (SmallCo), or U

(UnaidedCo) in Fig. 6) it was related to. The entries were sorted according to the relevant influence category. In Section 4 that presents the findings, Tables 7, 8, and 10 use the same format as was used for cross-case synthesis.

The process used for analysing individual cases is depicted in Fig. 7. All the collected data was imported into the qualitative data analysis software MAXQDA Pro 2020, and deductively coded within the coding structure shown in Table 5. According to (Maxwell, 2013), there are three main types of codes. Organizational categories represent areas you want to investigate. Substantive categories describe what happened or what was said. Theoretical categories place the data into a more general abstract framework. In this study, organisational categories were selected according to the research questions, as shown in Table 5. These categories were used to structure the data material, and initial coding used only these categories to organise the data. This could be considered indexing, in line with recommendations from Deterding and Waters (2021), and implied coding larger chunks of text into the organising categories. Then, for each of the organising categories, we performed analytical coding into substantive and theoretical categories. This is in line with recommendations from Deterding and Waters (2021) to focus on one research question at a time and to apply only a few analytic codes at the time to increase the reliability and validity of the coding.

As we were interested in identifying effects on the priority given to security (RQ1), we used previous knowledge on influences on security priority as theoretical codes (Tøndel et al., 2022). Thus, our coding approach was deductive. Our deductive approach was however not motivated by theory-testing (Wohlin

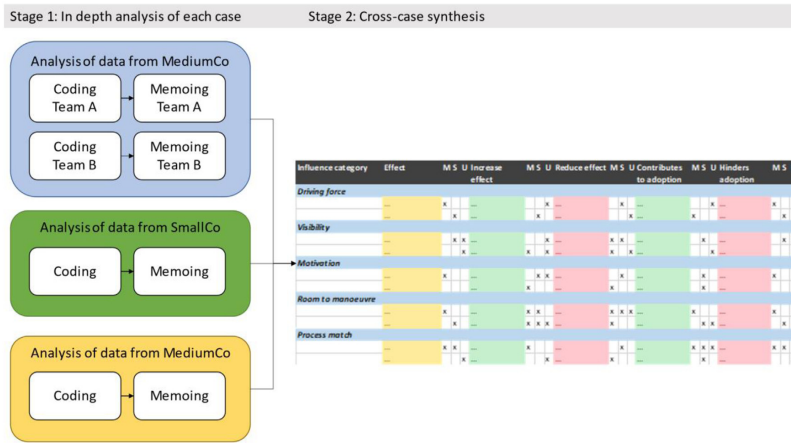


Fig. 6. Overview of the analysis process.

Table 5

Overview of coding structure.

Organisational category – area	Organisational category – subarea	Theoretical codes	Relation to RQs
Adoption	Examples of adoption	NA	RQ3
	Examples of non-adoption	NA	
	Reasons for adoption	Driving force; Visibility; Motivation; Room to manoeuvre; Process match	
	Challenges to adoption	Driving force; Visibility; Motivation; Room to manoeuvre; Process match	
Effects	Positive effects	Driving force; Visibility; Motivation; Room to manoeuvre; Process match	RQ2
	Contributes to effect	Driving force; Visibility; Motivation; Room to manoeuvre; Process match	RQ1
	– Context-related		
	– Meeting-related		
– Task-related			
Challenges and improvement suggestions	Hinders effect	Driving force; Visibility; Motivation; Room to manoeuvre; Process match	RQ1
	– Context-related		
	– Meeting-related		
Worked well	– Task-related		
	Challenges in the meeting	NA	RQ1
Worked well in the context	Challenges in the context	NA	
	Worked well in the meeting	NA	RQ1
Worked well in the context	Worked well in the context	NA	

and Aurum, 2015). Rather, it was motivated by a need to approach and understand this rather broad and abstract concept (the priority given to security) in a systematic way. For definitions of the influence categories, see Table 6. As can be seen from the overview of the coding structure in Table 5, we used these theoretical categories in the coding related to adoption and effect, based on the following considerations:

- Effect (RQ2): We had a special interest in identifying and understanding effects related to security prioritisation, and these influence categories could help identify effects likely to have an impact on the prioritisation given to security
- Aspects that contributed to or hindered effects (RQ1): The five categories structure influences on the priority given to

security, and thus could also help identify and structure influences on the effect of these meetings.

- Reasons and challenges for adoption (RQ3): We hypothesised that these influence categories could also help identify and structure conditions that influence adoption of the meetings, as adoption of these meetings can be considered part of giving security priority.

As was presented in Section 2 and summarised in Fig. 1, there are many potential effects and recommended strategies that support the suitability of regular security meetings to achieve prioritisations and decisions regarding security. The influence areas we decided to use as theoretical categories in the coding are part of this foundation. Still, it was important to ensure that we, by deciding to build on these influence areas, did not exclude



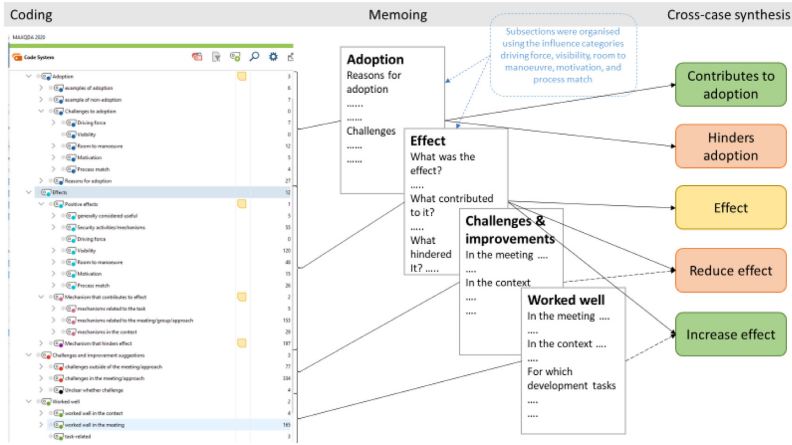


Fig. 7. Strategy for coding, memoing, and contributing to cross-case synthesis.

Table 6  
Influence areas from Tøndel et al. (2022).

Influence area	Definition from Tøndel et al. (2022)
Driving force	Someone who takes initiative and responsibility for making software security happen. A negative driving force would actively hinder software security.
Visibility	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	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	Resources and opportunities to prioritise software security, and to act accordingly. This might include time, budget, competence, etc.
Process match	The ability to fit the security approach into the existing software development process, so that they align well.

other aspects that could be important as well. We thus performed a mapping of the effects and recommended strategies identified from the broader set of literature (and as depicted in Fig. 1) with the influence areas from Tøndel et al. (2022). This mapping is shown in Fig. 8. Some of the relations are quite clear, like the relation between the effect “Visible and tangible security” and the influence area “Visibility”. Other relations were more subtle. Examples are the effect “Ongoing priority of security” and the recommended strategy “Context supportive of emerging security practices”. These we consider covered by all the influence areas in combination. Considering each influence area, *driving force* includes effects and strategies related to stakeholders as these can be important driving forces (or the opposite) for security prioritisation. *Visibility* includes making security more tangible, e.g., through prioritisation and documentation. *Motivation* includes getting towards an ongoing priority of security. *Room to manoeuvre* includes aspects related to reflection and learning, as this supports security knowledge and awareness. *Process match* concerns how the process for security prioritisation and follow up is organised, including who is involved.

After coding, we wrote four longer memos per case (as shown in Fig. 7), identifying and describing the findings related to the topic of the memo: adoption, effect, challenges and improvements, and worked well. Fig. 9 provides an example from this process. The memos offered an opportunity to summarise the key findings and reflect on them. The key findings were then again used as input to the cross-case table. As is shown in Fig. 7, the memos on adoption and effects were organised using the influence categories, and were used as the main input for the cross-case table. The memos on challenges and improvements and on what worked well were used to complement the findings.

#### 4. Findings

In the following we present the findings, organised according to the research questions. We start by describing the effects of the meetings (RQ2). Then we move on to presenting lessons learned on what contributed to or hindered the effect (RQ1). Finally, we describe findings related to adoption (RQ3).

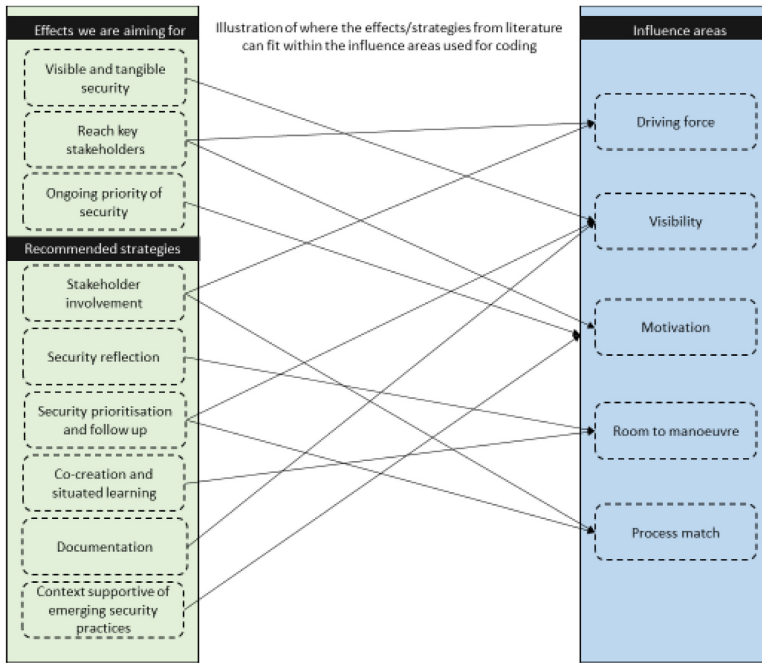


Fig. 8. The effects and strategies identified in literature (ref. the overview given in Fig. 1) can be covered by deductive coding based on the influence categories from Tøndel et al. (2022). Note that the effect 'ongoing priority of security' and the strategy 'context supportive of emerging security practices' are covered by all influence categories in combination.

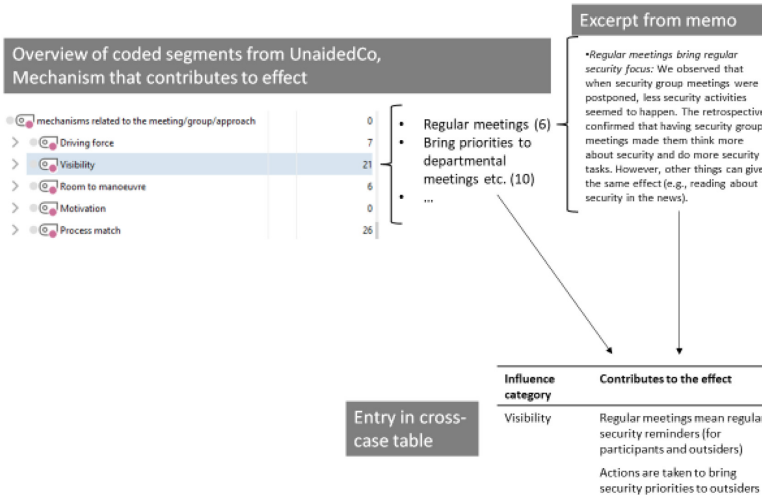


Fig. 9. Example illustrating the link between the codes, the memos, and the cross-case table.

**Table 7**  
Effects on the influence categories on security priority (M = MediumCo, S = SmallCo, U = UnaidedCo).

Influence category	Effect	M	S	U
Driving force	Enabling developers to take on responsibility and initiative for security in their work (including further security meetings)		X	X
Visibility	Identify, clarify, and document security needs, issues, and tasks	X	X	X
	Visibility of the security tasks they are doing, and the security they currently have in place		X	X
	Uncover non-functioning security roles/tasks	X		
Motivation	Positive view of the security work and meetings in the department/company	X	X	X
Room to manoeuvre	Awareness and knowledge building on security	X	X	X
	Reuse of knowledge and security work across projects, opening for reduced cost of security	X	X	X
	Increased confidence on security and on the decisions made, and opening for getting support for decisions from colleagues		X	X
Process match	A way to start and continue working with security, including ideas for how to modify and adapt the approach to better match their needs	X	X	

4.1. Effects from the meetings

The meetings brought many positive effects. All meetings led to the creation of security documentation. SmallCo and UnaidedCo reported on direct effects in their development; examples being a merge request template, security workshops, developing an incident response management process, starting to use a password manager, implementation of solutions for signatures and authentication, improved solution for remote support, the establishment of additional security meetings, and fixing of identified weaknesses. As shown in Table 7, the meetings also brought effects related to all the influence categories previously found to affect the priority given to security.

Looking at the influence categories, the main effects came within 'room to manoeuvre' and 'visibility'. In all three companies, the meetings helped build security competence and awareness among participants. The discussions brought both general and specific security competence relevant for the software being developed, and note-taking made the identified security needs, issues, and tasks visible also longer term. There is even evidence that this awareness, competence, and visibility spread to individuals who did not participate in the meetings (in the following termed 'outsiders'). As meeting participants gained more competence and confidence on security, they improved their ability to act as a driving force for security. Note, however, that the meetings did not necessarily give more time for security, although there is some evidence that they made it easier to ask for time to do security tasks (SmallCo).

The meetings helped get an overview of current security work and uncover potentials to improve this work, thus contributing to more cost-effective security. Improvements identified concerned reuse across projects, addressing non-functional security roles/tasks, and improving the security meetings. The meetings offered one way to get started with and continue working with security.

4.2. Lessons learned on meeting organisation

Table 8 gives an overview of identified influences on the effect of the meeting. In the following we describe lessons learned from all the cases when it comes to effectively organising these meetings. We start with describing the similarities identified across cases. Then, we bring up the main variations among the cases. Finally, we delve into one overarching issue emerging from the analysis: confidence in the software security prioritisations and their effect.

4.2.1. Similarities across the cases

In all three cases, the following aspects of the meeting were important contributors to the effect:

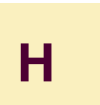
- A view in the company of security as important and worthwhile (motivation)
- Regular security meetings as regular security reminders towards both participants and outsiders (visibility)
- Participants positioned to take action and bring a security mindset to outsiders (driving force)
- Participants being positive and engaged towards the meeting and security (motivation), having security competence and experience (room to manoeuvre)
- Good discussions in the meetings to build awareness and competence (room to manoeuvre)
- Concrete action points from the meeting (process match)

These contributors could, e.g., play out as follows. UnaidedCo had for several years experienced an ongoing push for security and this ensured an opening to spend time and money on security (including gathering senior people for a security meeting). In UnaidedCo, participants explained that the meetings helped them think more about security and made them do more security tasks, and we observed that when the security meetings were postponed less security activities seemed to happen. In MediumCo, personal engagement motivated a product owner to take on responsibility for security tasks despite strong time pressure. Participants in UnaidedCo stated that the discussions were the most important part of the meeting. In MediumCo, one of the product owners held clear action points as the most useful result of the meetings.

All the studied cases experienced the following challenges in getting effect from the meeting:

- Important discussion points risked being lost as not all points seemed to be noted down (visibility)
- A focus on functionality in the company and among customers pushed security to the background (motivation)
- Time pressure made it hard to take on responsibility for and/or perform the action points, or made it hard to set aside time for necessary security training (room to manoeuvre)

In all meetings, one person took responsibility for taking notes, and usually these notes were made visible to all participants. Still, we observed that there was a risk of forgetting to write



**Table 8**  
Overview of influences on the effect of the meeting (M = MediumCo, S = SmallCo, U = UnaidedCo). Note that the term “outsiders” is here used to refer to company employees not participating in the meeting.

Influence category	Contributes to the effect	M	S	U	Hinders the effect	M	S	U
Driving force	Participants that are positioned to take action and bring a security mindset to outsiders	X	X	X				
	Facilitator that pushes for documentation and follow up of action points	X		X				
	Skilled security expert as facilitator and contributor to the meeting	X	X					
Visibility	Regular meetings mean regular security reminders (for participants and outsiders)	X	X	X	Important discussion points can be lost as not all points seem to be noted down	X	X	X
	Template gives visibility to topics for discussion and support in identifying and documenting issues	X	X		Outsiders may not read security documentation although it is made available to them		X	X
	Template brings attention to similarities and differences among projects	X			Outsiders do not feel they get enough information from the meeting			X
	Actions are taken to bring security priorities to outsiders		X	X	Visible costs of security while effects of security work are less clear			X
	Security issues visible to outsiders are easier prioritised			X				
Motivation	Participants that are engaged and positive towards the meeting and security	X	X	X	Focus on functionality in the company and among customers	X	X	X
	A view in the company of security as important and worthwhile	X	X	X	Too many prioritised tasks, long list of issues to address and consider			X
	External pushes for security, e.g., pentest, customers			X	Security tasks that are boring or unpleasant			X
Room to manoeuvre	Meeting participants with security competence and experience	X	X	X	Lack of trust in own judgement – analysis paralysis		X	
	Good security discussions in the meeting build awareness and competence	X	X	X	Challenges in understanding security terms in the template		X	
	Template (security areas, questions) that support discussion and knowledge building		X		Knowledge needs related to practical security solutions make it hard to translate decisions into code, etc.		X	
	Initiatives to bring security competence to the company		X	X	Time pressure makes it hard to take on responsibility for and/or perform the action points	X		X
	Room to spend time on security			X				
	Ability to identify a wide variety of issues quickly in the meeting	X			Tasks that are difficult, large, or concern old systems			X
			X		Hard to set aside time for training		X	
Having concrete things to discuss helps justify the time spent in the meeting				Lack of roles such as sysadmin to establish security infrastructure, etc.		X		
Process match	Concrete action points from the meeting	X	X	X	Not well placed to deal with non-project related issues	X	X	
	Ability to include results from meeting in external planning process			X	Participants that are not the right ones to be responsible for an action point	X		X
	Small company, low security maturity, easy to get effects		X		Need perspectives from outsiders	X		
					Lack process for following up meeting results in development	X		
				Meeting too early or too late during development	X	X		

down discussion points, and that it could be challenging to know what to document and how. Unsurprisingly, all studied companies experienced a push for functionality that impacted the priority of security. UnaidedCo explained that the product owners were mainly concerned with functionality. Similarly, MediumCo explained that customers pay for features, not security. Consequently, it was hard to push for security when the security tasks could delay development of features. This push for functionality was somewhat related to time pressure, which was particularly

strong in MediumCo. Their most stated reason for skipping security tasks was time pressure, and all roles experienced such pressure. One product owner thus explained that the security meetings mainly served to give him bad conscience, as it made him aware of all the things he did not manage to do:

*“Product owner: It kind of works that you are part of discussions and contribute with what you know, one hour and every month. However, it does not happen that much in-between.”*



*Interviewee: But it does not have an effect that you are reminded of it every month?*

*Product owner: Yes, reminds me of it and gets a bad conscience for everything that should be in Jira, and tasks related to that”.*

#### 4.2.2. Variation: participants

The characteristics of the meeting participants varied across the cases, as shown in Table 9. The cases had different needs to be met by the meetings. This can explain the variations, as all have their benefits and challenges. In the following we point to lessons learned when it comes to meeting participants.

It was important to have security competence in the meeting (room to manoeuvre), but it was not necessary to *only* have participants with security competence. To illustrate, in SmallCo the security competence was held by the facilitator who could explain terms, point to potential challenges, question assumptions, and point towards solutions. Further, the need for security competence did not mean there had to be a security expert in the meeting; many of the participants with developer, architect, or product owner roles had sufficient competence to identify and discuss security issues and mitigations. We, however, observed variations in depth and speed of discussion that may be related to security expertise and driving force. The clearest example of valuing efficiency over depth was found when bringing the meeting to a new team at MediumCo. Here the product owner leading the meeting managed to go through the full template, including all the 13 security areas, in 75 min. The previous meetings in MediumCo that were facilitated by a security expert, had spent considerably more time on each security area, digging deeper and asking hard questions on assumptions.

The meeting needed participants positioned to take responsibility for the tasks prioritised in the meeting (process match, room to manoeuvre). In UnaidedCo, meeting participants were given responsibility for security action points, while other tasks were normally the responsibility of team leads. Thus, the security tasks were not fully integrated in their process. However, assigning responsibility for security tasks to the team leads was challenging as they were not participants in the security meetings, and thus not present to report on the status of the action points. For MediumCo, meeting participants lacked the capacity to take on responsibility for more tasks due to time pressure. Thus, there were discussions on whether to add participants who were better positioned to do the necessary work on the action points (e.g., a security champion or developer). Further, many of the meeting discussions in MediumCo covered topics that involved operations or other development teams. Thus, participants lacked knowledge to make realistic assumptions about the risk, and many of the issues and action points identified were concerned with gathering more information.

#### 4.2.3. Variation: meeting scope

The meetings we studied either had a department scope or a project and team scope, and both scopes had their benefits and challenges (primarily related to process match). Regarding benefits, the department-scope of UnaidedCo made their meetings well placed to make decisions that affected the whole department and not only one project or team. Several department level initiatives stemmed from these meetings, including a merge request template and hacker workshops. The project-scoped meetings of MediumCo and SmallCo were able to dig deeper into the individual projects, but also cross-cutting security concerns were discussed. Although the department-level meetings were well positioned to support learning across projects and technology, such effects were also seen in MediumCo where cross-team learning happened through participants being involved in security work in several projects. And as stated by the developer in SmallCo:

*“We have talked about one project, but I have always kept in mind all the other projects”.*

Challenges were more prominent in the project-scoped meetings. Both scopes experienced issues falling between two stools; security issues could concern another development team (MediumCo), operations (MediumCo), or be too big to address within current plans (UnaidedCo). However, project-scoped meetings had more challenges in acting on cross-cutting concerns. Both scopes experienced challenges related to keeping a lifecycle perspective; also UnaidedCo found it harder to make security happen in existing vs. new systems. However, a product owner at MediumCo advocated for a product scope rather than a project scope in the meetings, as software changes were made also for products not in active development in a project; *“A customer comes and wants to pay 50 000 NOK to add a button, and then we add that button. This does not become a project, and then there is no security decisions meeting (...). We make a change; we spend two weeks and make a change and that’s it”.*

Challenges related to meeting scheduling were specific for project-level meetings. When scheduling meetings too early in the project, there was not enough information to make security prioritisations and decisions. Changes to projects were normal, especially in the beginning, thus there were considerations on how long to wait for things to settle. Further, there were concerns about when to revisit previous assumptions. When starting the meetings too late, the option to influence the project plans and estimations were largely lost. We observed a risk that the meeting could be experienced as a security kick-off for a project, without this leading to regular security meetings as was the intention.

#### 4.2.4. Variation: support material

Both SmallCo and MediumCo used support material in form of security areas and security questions in their meetings (see Appendix A). Observations pointed to a potential effect in both companies, related to triggering ideas for discussion, aiding in documentation, supporting awareness and knowledge building, and building confidence in the assessments made (visibility, room to manoeuvre). The material supported the experienced facilitator, and even allowed the product owner in the new team at MediumCo to run the meeting after being introduced to the material. In SmallCo, it offered a way to get started with the big topic of security, by breaking security into more manageable pieces, and it brought visibility to topics that had not been much considered in their solutions before; *“The biggest advantage is that you know a bit more, it is more structured what to talk about. It is easier to remember what we have talked about, and what we have not talked about”* (developer SmallCo). These effects came despite both companies identifying many potential improvements to the material. Note, however, that UnaidedCo did not use such support and still covered a broad set of security aspects in the discussions, produced organised security documentation, and built competence and awareness on security.

#### 4.2.5. Variation: company’s size and security maturity

Our study included a very small company just starting to work with software security (SmallCo), as well as medium-sized companies that already had experience with software security (MediumCo and UnaidedCo). SmallCo experienced challenges related to room to manoeuvre, while MediumCo and UnaidedCo experienced challenges related to process match and visibility.

Competence was important in all cases, but as a small company with few developers and technical resources, SmallCo experienced that a lack of practical security skills was a hindrance for implementing security measures. Further, as they lacked roles such as sysadmin, developers had to take broader responsibility for practical tasks. In, e.g., MediumCo, such roles were filled,



**Table 9**  
Observed variations that can be related to the selection of participants.

Case	Participants	Benefits	Challenges	Main effect
MediumCo – initial project	Small set of participants with decision making power and competence. Facilitation by security expert.	Good discussions, confidence.	Individuals highly pressured for time, lacked involvement in all parts of the project.	Addressed the need to identify security issues not explicit in customer requirements and get towards solving the security issues.
MediumCo – new team	Full team. Facilitation by product owner.	Broad awareness of security issues.	Efficiency over depth in discussions. Did not redo the meeting.	Identification and documentation of issues.
SmallCo	Two developers. Facilitation by security expert.	Developers that could bring improved security focus.	Need additional meetings to bring broader changes.	Got started, built competence, established new practices.
UnaidedCo	Seniors. Facilitation by manager.	Reach key actors in the department, positioned to make changes.	Get broader effects, reach beyond the participants, get tasks prioritised by team leaders.	Identified, documented, and addressed issues. Support for participants in their ongoing attention to security.

and more competence was available, but the challenges were related to company silos and security concerns being viewed as part of someone else’s responsibility. To exemplify, the new team applying the meeting found that for many issues they were dependent on third parties, other teams, or operations. However, these issues were generally skipped in the discussions, thus no action was made to ensure that they were in fact addressed.

Integration into the larger development process was challenging for the medium-sized companies. In MediumCo, the product owners were used to doing refinement in Jira. There was, however, no surrounding security process that ensured action points from the meetings were followed up on, e.g., by adding them to Jira. In UnaidedCo, security tasks were generally not included as user stories and added to Target Process and their Kanban. Thus, in both companies there was a need to remember to look at a separate list/page for security tasks. Adding the security tasks into Jira or TargetProcess was, however, not without challenges. In MediumCo, Jira was explained as already being filled with too much information, making it hard to navigate. It was thus easier to get an overview of all security decisions in a Confluence page. In UnaidedCo, adding all security concerns to Target Process was not an option, as the information was considered too sensitive to store in a Cloud solution. Note that, on a related point, UnaidedCo successfully included larger security tasks in yearly plans for the department, showing that integration with the planning process already in place could be effective.

Both SmallCo and UnaidedCo took action to spread information from the meetings to outsiders. In SmallCo it was easy for the participants to discuss the meetings informally with other employees, including management. Still, they started a new security meeting series with management. UnaidedCo spread information from the meetings in weekly department-level status meetings and in emails – with meetings being most effective. Still, outsiders expressed that they wanted more information from the security meetings, and roles outside the department (such as product owners) most likely did not know about these meetings and the prioritisations made there. Broad sharing of information was however challenging, as much of the security documentation created in the meetings was considered highly sensitive. Further, making security documentation available to a broader set of individuals did not imply that this documentation was read and understood by others.

4.2.6. Variation: maturity of meeting series

Challenges to the meetings varied depending on whether the meeting series was just starting, or whether it had been going on for some time. Initial challenges included understanding

how much time to set aside, and clearly communicating the goal and structure of the meeting (MediumCo). Further on, challenges included how to proceed when all security areas had been discussed, and when to revisit assumptions (SmallCo). After meetings had been going on for a long time, the number of issues identified but not yet addressed could be challenging to manage (UnaidedCo) – as stated in the retrospective of UnaidedCo: *if you should go through the to-do list, this is the whole meeting.*

4.2.7. Overarching issue: confidence in the software security prioritisation and follow-up

Experiences from UnaidedCo showed that prioritisation and concretisation of tasks were important prerequisites for action. Security discussions took place also before they started with this meeting series, but those discussions usually did not lead to actions, as there was no clear process for following up on the issues. Due to the meeting series, all these issues were documented, and they ended up with a long list of security issues. To start addressing the issues, prioritisation became important. But prioritisation was also challenging. The retrospective showed that often more action points were prioritised than what was realistic to address before the next meeting, leading to erosion of responsibility.

Challenges related to prioritisation were found within all influence categories. In all the studied cases it was difficult for the observer to understand why some issues were prioritised over other issues, indicating unclear prioritisation criteria. The developer from SmallCo talked about the risk of analysis paralysis, especially if participants lacked confidence in own ability to make good security decisions. For UnaidedCo, it was challenging to manage the long list of concerns identified over the course of all meetings, and thus it seemed easier to prioritise newly identified concerns. Furthermore, tasks that were boring (e.g., fixing an existing system) or unpleasant (e.g., could cause down-time or required work outside of normal working hours) were less likely to be prioritised.

Prioritisation also happened outside of meetings, and in relation to the totality of the tasks that the participants were expected to address. This prioritisation (security vs. features) was described as more challenging than the prioritisation among security tasks happening in the meetings. In the meetings, we observed that when action points had not been addressed, there was often little discussion as to why this was the case. Thus one missed the opportunity to learn about barriers to security work and improve how action points were addressed in the future. All observed meetings benefited from an open and non-judgemental tone where participants were willing to share knowledge needs and insecurities. However, the need to hold participants account-





able for following up on their responsibilities for action points was slightly neglected.

#### 4.3. Conditions leading to adoption

An overview of what contributed to or hindered the adoption of the meeting, both shorter and longer term, is given in Table 10. In the following we describe mainly what we found to be important for longer-term adoption. The meetings that were adopted longer term were the monthly security group meetings at UnaidedCo, that had been going on for some time before our study, and the monthly management meetings on security, that were started by SmallCo while we did our study with them. Further, SmallCo included security in daily meetings among developers. The meetings we initiated in SmallCo and MediumCo were not continued.

The meetings that were adopted longer-term shared some characteristics. Though both SmallCo and UnaidedCo were triggered by external security experts in initiating their meetings, the meeting approach they applied had been created by the company itself, and was driven by key individuals from development. Both companies applied cross-project meetings. Moreover, management in both companies acknowledged the importance of spending time on security, and the time needed for the meetings was perceived as acceptable.

Offering a good process match was supportive of adoption, although not enough to ensure or hinder adoption (process match could be achieved over time). It was considered beneficial to have an easy approach that could be done efficiently, and that could be adapted to the needs of the company. On the other hand, it was challenging with security meetings that were somewhat “on the side” of their development process, and thus had to be remembered and prioritised over “real development tasks”, with each development project needing to consider when to start with security meetings.

In both cases where we brought in the support material and helped facilitate the meetings, these ended up not being adopted. The reasons put out were limited need for such a thorough approach in the new projects they had currently initiated (SmallCo), and time pressure of product owners in combination with no central security officer role that continued to push for the meetings (MediumCo). These are all aspects of the context. Thus, we cannot conclude that aspects of the support material or the agenda of these meetings prevented adoption. On the contrary, SmallCo expressed an intention to continue using this support material; *that excel sheet was really good, so we need to remember to use that!* (statement from observation notes).

## 5. Discussion

We started this article with introducing the concept of continuous software security. This concept implies that software security is treated as a key concern throughout the software’s lifecycle (Fitzgerald and Stol, 2014). For this to happen, literature points to the need to reach key stakeholders with software security, to make security more visible and tangible, and to prioritise security in an ongoing manner (Fig. 1). In this section we relate our findings to the literature, and identify implications for research and practice. We organise the discussion according to the topics we identified from literature in Fig. 1, and we use **bold** whenever we refer to topics in that figure.

The meetings contributed towards **ongoing priority of security** directly through the activities that happened in the meeting, and through positive effects within all the influence categories related to security priority (Table 7). This included contributions towards **visible and tangible security**. The effects observed resemble those found for related techniques (Fig. 2) in that the meetings led to concrete security improvements, and the strongest effects were related to visibility, competence, and awareness of security. Thus, we claim that such effects can be expected from security meetings in general. However, literature points to stronger effects of security workshops in smaller companies, compared to larger and more mature ones (Weir et al., 2021). In the study by Weir et al. (2021), this finding may, however, be due to organisational turmoil in one of the large companies they studied, rather than their approach (Weir et al., 2020a). We found that the meetings were effective in all cases, but that it was easier to get effects in the smaller and less mature company (SmallCo).

#### Implication for practice:

(P1) Regular security meetings are recommended for small and medium sized development companies that need to strengthen their software security maturity.

#### Implication for research

(R1) As we found that regular security meetings can be effective in smaller companies, further research should study what role (if any) regular security meetings can play in larger and more mature organisations, and how they should be organised in these contexts to support adoption and be effective.

The meetings’ ability to **reach key stakeholders** was related to who was participating in the meeting, although we experienced that the effects of the meeting could reach beyond participants. Relevant strategies when deciding on participants are **stakeholder involvement** and **co-creation and situated learning**, but these need to be balanced towards the concern that larger meetings tend to be less effective (Stray et al., 2016). The variations among the cases concerning meeting participants, all had their pros and cons (Table 9), and it appears that there is no one-size-fits-all in this respect, as different participants may serve diverse needs (Weir et al., 2020a). Recommendations on whether to include the full team, a security expert, managers, and product owners come up in literature (see Fig. 2). Experiences from our study relate to these recommendations in the following way:

- Literature highlights the importance of involving the full team and not only seniors (see Fig. 2). Both MediumCo and UnaidedCo violated this recommendation, and still found the meetings effective. However, we found that the meeting needed some participants with room to take on responsibility for security tasks, and seniors may experience more time-pressure hindering them to take on this responsibility.
- Literature conflicts on whether meetings need a security expert (Weir et al., 2020a; Bernsmed et al., 2022). We found involvement of a security expert to be beneficial, and probably necessary when initial security competence was low (as in SmallCo). However, a security expert was unnecessary when at least some participants were aware and knowledgeable about security.
- Previous findings that facilitation by management is beneficial (Weir et al., 2021) is somewhat supported (UnaidedCo).

- The importance of product owner participation is unclear in literature and in our study. Weir et al. (2021) found surprisingly few effects of involving the product owner. In our study, product owners were involved in MediumCo but not in UnaidedCo. Both report on challenges that functionality is prioritised over security, and one of the product owners of MediumCo expressed that the meeting mainly led to bad conscience, not improved security.

**Implication for practice:**  
 (P2) When selecting participants, consider what are the main needs for your team/project/company. If you are in dire need of security competence, consider involving a security expert with the full team, alternatively a security expert with individuals interested in security, who can spread security competence and awareness to their team (e.g., security champions). If you need decision making power and competence, consider involving senior people. If communication and overview is a main need, consider involving participants across silos (e.g., both from dev and ops).  
 (P3) Include participants who can take responsibility for security tasks in development.  
 (P4) If possible, have a manager as facilitator of the meetings.

**Implication for research:**  
 (R2) As our results suggest that there is no one-size-fits-all regarding meeting participants, companies need guidance on which participants to include for varying meeting aims and contexts.

Weir et al. (2020a) recommended “the promotion of software development security as a systemic, rather than purely a development team, matter”. Literature shows that awareness of security weaknesses is not enough to induce change (Palombo et al., 2020). Clearly, a meeting alone cannot create a **context supportive of emerging security practices**. It does not replace the need to address the more structural and systemic blockers that hinder developers and product owners in prioritising security in practice, but it can support identification of these blockers. For this, product owner participation appears important but not necessary; discussions of systemic blockers and conflicting demands took place in all cases. However, these were often hard to address in practice and they could easily be viewed out of scope for the meeting, especially if the meeting had a project scope.

**Implication for practice**  
 (P5) Product owners can have an important role to play in the meetings, but beware that meeting participation will not necessarily change the priority they give to security if systemic blockers are not addressed.  
 (P6) Discussing reasons for not doing security work (without assigning blame), is one potential way to get more insight into the blockers that are present.  
 (P7) If there is a strong need for overarching changes, a department-level meeting may be called for.

**Implication for research**  
 (R3) Systemic blockers for software security came up in meeting discussions, indicating that security meetings can address such blockers. Still, more knowledge is needed on how to position and structure security meetings to best address systemic concerns, including how to document and follow up on such concerns within and beyond the meetings.

Literature advocates for emerging security practices (in contrast to prescribed practices) (Türpe and Poller, 2017; Weir et al., 2020a). Our study support this; the practices that are adopted longer term are those that emerged in the companies. Further, we hypothesise that emerging practices can be better positioned to deal with challenges in the context that may hinder security work, such as time-pressure – a prominent challenge in literature (Fig. 2) and in our study. The studied companies were able to select a meeting schedule that suited their need, and thus time for the meeting seemed not to be a main issue, although meetings were sometimes postponed. To ensure continuous adoption, we would highlight the importance of having a strong driving force for the meeting, someone with the authority to invite the right participants, facilitate the meeting, and ensure meetings are arranged regularly. We identified a potential need to change the meeting as a project progresses (SmallCo) and as the number of identified concerns increases (UnaidedCo). Although findings suggest that a good process match is supportive of adoption, this can be achieved over time. An engaged facilitator can take responsibility for making strategic decisions on how to improve the meeting, and adjust to changes in the needs of a project and/or company.

**Implication for practice:**  
 (P8) Adopt your own meeting approach that suits your needs (including your level of time-pressure), rather than copying an approach from others. This does not exclude learning from others’ experiences.  
 (P9) Ensure that someone is championing the meeting, and that this person has the necessary authority and motivation to make the meeting happen regularly and make improvements.

**Implication for research:**  
 (R4) As results indicate that meeting effectiveness can change with time, companies can benefit from guidance on how to ensure meeting adoption and efficiency in varying stages (e.g., as a project progresses or as the company matures).

The scope of the studied meetings varied, with some taking a project-scope and others a department-scope. We cannot claim that one is always better than the other, although some benefits were identified with a department scope (stronger ability to address more overarching concerns and cover products not in active development). We also saw a potential challenge with project-scope meetings in that each project must remember to initiate its meeting series.

**Implication for practice:**  
 (P10) If you go for a project-specific security meeting, ensure that the meeting becomes part of routines so that the meeting is remembered for all projects. Further, consider whether there is a need for meetings also for key products not under active development.

**Implication for research:**  
 (R5) Our results slightly favour department-scoped meetings, and we speculate that this is related to us studying smaller companies where such broader-scoped meetings may be more feasible than in larger companies. More knowledge is needed to determine what meeting scope is most effective in varying contexts.



In the meetings, time for **security reflection** and discussion was highly important for the effect of the meetings. This is in line with experiences from similar techniques (Fig. 2). The effect was however linked to **documentation**; there was a need to ensure key discussion points were documented (and not lost Cruzes et al., 2018; Tøndel et al., 2019b), and visible documentation supported the discussions (Stray et al., 2016). This study is not conclusive on how to structure the discussions and the documentation. Meetings were successful both with and without support material. Support material in form of a checklist has been suggested as an improvement to the security workshops studied by Weir et al. (2020a). The main benefits of the support material as used in this study was that it made security more concrete and manageable, structured the discussions, gave confidence, and supported non-experts as meeting facilitators. However, we are not aware that the companies continued using it after the study.

**Implication for practice:**

- (P11) Meetings should allow ample time for discussions; thus, the agenda should not be too rigid.
- (P12) One participant should be responsible for taking notes.
- (P13) Notes on a shared screen, including notes from previous meetings, can support discussions.

**Implication for research:**

- (R6) Though this study sheds light on the potential benefits of using support material in security meetings, the necessity of such material is not clear. The potential role of support material should be investigated further, including when such support is most needed, what support is effective for which types of meetings and contexts (e.g., with a department-scope vs. a project-scope), and how the needs for support material can vary with time as the company's security maturity changes.

Positive effects of the meetings were observed in all the studied cases, despite all cases struggling with how to get the most effect from the meetings. We even discovered a potential for the meeting to contribute with more cost-effective security, e.g., through supporting reuse across project and address non-functional security roles and tasks. **Security prioritisation and follow up** was essential for getting these positive effects. Still, confidence in the decisions was challenging, and it was not always clear how the security priorities were made. Similar challenges have been identified also for other security meeting types (Fig. 2).

Studies show that not all security vulnerabilities are exploited (Nayak et al., 2014). Thus, to arrive at cost-effective security, it is important to address those issues most likely to cause problems. Yet, we observed that questions like “what is most important?” often remained unanswered in the meetings, whereas other criteria (e.g., ease, concreteness) tended to be more used in the prioritisation.

A learning point from the meetings was that it was important to spread information to outsiders. Prioritisation of security did not only happen in the meetings – the prioritisation of security vs. features and other tasks that happened outside of the meetings was even more challenging than the prioritisation that happened in the meeting.

**Implication for practice:**

- (P14) Beware of the tendency to prioritise easy and concrete tasks and newly identified tasks, without considering whether they are the most important tasks.
- (P15) Beware of prioritising too many action points in a meeting. It is better to identify a few action points that end up being addressed, than identifying many action points where the understanding is that all will not be addressed.
- (P16) For those action points that are prioritised, there is a need to ensure that they are concrete, and that responsibility and deadline are properly defined.
- (P17) As the number of identified concerns grow, consider pruning the list outside of the meeting, to avoid overload and make the meeting more engaging.
- (P18) Actions should be taken to bring information from the security meetings to a broader set of individuals. Sharing of such information in person (e.g., in meetings rather than on email) is preferred.

**Implication for research:**

- (R7) As results indicate that security meetings can contribute to more cost-effective security, future research could investigate how meetings can be organised to support cost-effectiveness and how this cost-effectiveness can be assessed and made visible.
- (R8) Research can contribute with prioritisation and decision-making support and strategies to be used in security meetings, to address the overarching challenge of confidence in security prioritisation.
- (R9) Companies can benefit from improved strategies for communicating priorities from the meeting, and integrating them into their way of working.

**6. Threats to validity**

In the following we discuss the threats to the validity of our study results, using the classification scheme suggested for case studies (Runeson and Höst, 2009; Yin, 2018).

*Construct validity* can be defined as the “accuracy with which a case study's measures reflect the concepts being studied” (Yin, 2018) (Runeson and Höst, 2009). The concept of security priority is important in this study; we were interested in understanding the effects these meetings had on security prioritisation. ‘Security priority’ was operationalised through five influence categories: driving force, visibility, motivation, room to manoeuvre, and process match (Tøndel et al., 2022). These influence categories stem from one case study, and thus cannot be said to be an established theory on what brings priority to security in ASD. We are however not aware of the existence of such a theory. Though stemming from one case study, these influences are prominent also in the broader literature (Tøndel et al., 2022), strengthening our assumption that they would be relevant also in the contexts studied in this paper. We did find these influence categories useful in structuring and reasoning about our findings, something that supports their relevance. Still, there is a need for more studies that can form a stronger theoretical basis for understanding what brings priority to security in ASD. Measuring the effect on security prioritisation was however challenging, also with the use of these influence categories. Effects could be quite invisible (e.g., knowledge, motivation) and they could manifest outside of the meetings. Thus, we relied on participants reporting this effect in meetings or in interviews/retrospective.



*Internal validity* is concerned with causal relations and the risk that effects observed may have been caused by factors not considered in the study (Runeson and Höst, 2009). In qualitative studies, internal validity can be supported by strategies such as triangulation, thick descriptions, linking results to theory, seeking negative evidence, considering rival explanations, and having participants find the conclusions accurate (Miles et al., 2018). These strategies were applied in this study. Still, there are potential biases. The influence categories used in the analysis helped relate to the broad concept of security prioritisation. However, by building this strongly on these influence categories, that we ourselves had developed in a previous case study, there is a risk of confirmation bias. This study did not use these influence categories in order to confirm them. Still, the categories informed our understanding of security prioritisation and may have led us to see prioritisation that was not there, or miss aspects of prioritisation that we had not previously identified. However, these risks would be there also with a more inductive analysis approach, although less visible.

We acknowledge that in the cases where we used action research, our influence as researchers was considerable (Petersen and Gencel, 2013). We helped develop the meeting used in SmallCo and MediumCo, and thus there is a risk of bias related to us wanting this meeting to succeed. We have been aware of this risk throughout. Also note that the findings tip in favour of department level meetings (which we did not initially suggest). It is likely that our presence as researchers influenced the meetings we studied. Participants may have wanted to let the meetings we facilitated (MediumCo, SmallCo) look good to please us as researchers. They may also have wanted the meetings they organised (UnaidedCo) to come out as successful. We took care to regularly reflect on our influence as researchers, as part of the observation template. In interviews we made sure to express a need to not only hear about the good aspects of the meetings, but that we wanted to know about the challenging parts as well, so that we could improve. Our impression is that participants trusted us enough to give us their honest feedback. For us, it was particularly important that participants were honest about what they saw as effects of the meeting, as we did not have direct ways to measure this. In addition to encouraging participants to give honest feedback, we considered rival explanations in our analysis. To exemplify, in the study we saw that it was easier to get and see the effects of the meeting when you started from “nothing” (as in SmallCo). But then, what we saw as effects might not be effects of the meeting approach but rather effects of starting to work on security and to interact with external security experts. Thus, we were restrictive in claiming something to be effects of the meeting and made efforts to point to contextual factors that could be important for the effects.

*External validity* “is concerned with to what extent it is possible to generalise the findings, and to what extent the findings are of interest to other people outside the investigated case” (Runeson and Höst, 2009). As is common for case studies and action research, there are many contexts and many meeting approaches that we have not considered in this study. However, studying three companies with varying meeting operationalisations, allowed for identifying similarities. Thus, it pointed us towards findings that are likely to be shared by more than one context/meeting type. To support readers in judging whether the findings might be relevant for their context, we provided details on the company contexts and the meetings we studied (within

the limitation that the anonymity of the companies should not be compromised). Note that the companies we have studied are of small or medium size, and we expect that results for larger companies may vary considerably from the findings in this study.

*Reliability* “is concerned with to what extent the data and the analysis are dependent on the specific researchers” (Runeson and Höst, 2009). In our study, reliability is supported through the use of an observation template, the recording and transcription of interviews, and the clear protocol for analysing the data. Further, data triangulation and member checking (as part of interviews, retrospectives, and the sharing of the draft research report) help support reliability. To exemplify, in interviews and retrospectives we took care to let the participants present their view, without first bringing in our understanding. Still, we also used the opportunity to provide our understanding when relevant, and get the interviewee to respond to that. We also shared a previous draft of this article with key study participants, and the feedback received support the validity of the findings. In the action research part of our study, two researchers were involved, where one took the main role as facilitator and the other did the data collection and analysis, thus being able to take a more external view of the meeting. Having only one researcher doing the analysis represents a threat to reliability. However, findings were discussed among the two researchers at several points throughout analysis. Replication is a general challenge for action research studies as it relies heavily on a trust relationship between the researcher and the company (Wieringa, 2014). However, to support replication in theory, we have provided details about the data collection instruments and the meeting concept used, as well as our approach to analysis.

## 7. Conclusion

This article proposes regular security meetings as a strategy for continuous software security, and reports on a study of such meetings in three companies. The studied meetings varied in scope (project/team or department), participants (full team, seniors), support material, and context (small to medium size, low to medium security maturity).

Results from this study show that regular security meetings can contribute to ongoing priority of security, more visible and tangible security, and can reach key stakeholders — all these effects are called for in literature. Based on the lessons learned from the cases, we identify implications for practice and for research.

For practitioners, we find evidence that regular security meetings are useful for small and medium sized development companies that need to strengthen their software security maturity. Companies should seek to adopt a meeting approach aligned with their own needs, this includes selecting a meeting scope, participants, a meeting schedule, and a meeting structure. However, the lessons learned from this study can give some directions, e.g., pointing out the need for participants able to take responsibility for tasks, the importance of having time for discussions, and benefits and pitfalls of a department scope vs. a project scope.

For future research, we point to the need for more competence to improve the ability to support companies in selecting a meeting approach that suits their needs. This includes knowledge on how meetings should be adapted to different contexts and needs, knowledge on how to position the meeting to address systemic blockers for software security, and knowledge on the role of support material in different meeting models.

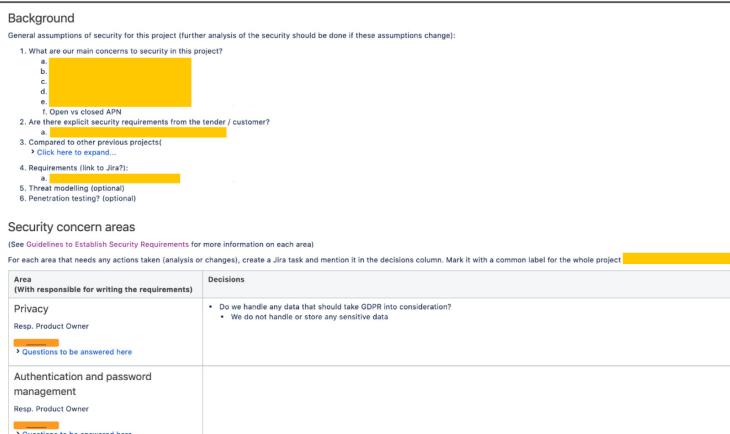


Fig. 10. Excerpt from Confluence page of MediumCo.

**Table 10**  
Influences on adoption (M = MediumCo, S = SmallCo, U = UnaidedCo).

Influence category	Contributes to the adoption	M	S	U	Hinders the adoption	M	S	U
Driving force	Someone with authority initiating and inviting to meetings	X	X	X				
	Participants are senior people			X				
Visibility								
Motivation	Participants that are motivated to meet and discuss security	X	X	X	Not all projects have a clear security need, thus meeting not always necessary			X
	A general push for security in the company		X	X				
Room to manoeuvre	The cost of security meetings is accepted			X	Challenging to set aside time	X	X	
	The meeting is considered easy to do, and can be done quickly	X	X					
Process match	Ability to adapt the meeting to own needs		X		Disconnected from the “real work” they are doing that is more urgent	X		
	The practice of deciding on a time for the next meeting	X	X		Initial project work is creative and exploring, not considering security, etc.			X
					If it turns out that all projects have similar security needs, a checklist may be a better option			X

**CRedit authorship contribution statement**

**Inger Anne Tøndel:** Conceptualization, Methodology, Validation, Investigation, Writing – original draft, Visualization. **Daniela Soares Cruzes:** Conceptualization, Methodology, Validation, Resources, Writing – review & editing, Funding acquisition.

**Declaration of competing interest**

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: The authors have previous relationships with two of the companies, either in form of previous part time deployment or being involved in consultancy projects towards companies.

**Acknowledgements**

This research was funded by the Research Council of Norway, grant nr. 247678. The authors would like to thank the companies where this study was performed, especially the interviewees and the participants in the meetings we observed. Thanks to our colleague Martin Gilje Jaatun for participating in the data collection at UnaidedCo. Thanks to Prof. Guttorm Sindre and our colleagues in the “Fabrikk” for many useful comments on this article.

**Appendix A. Meeting approach used by the companies**

The meetings in UnaidedCo had the following structure. In the beginning of the meeting, they went through the list of activities that was prioritised in the previous meeting, to assess status. In

**Table 11**  
Overview of areas covered by the support material, as used in SmallCo.

	Area	Supporting questions
Overall concerns	Level of security we aim for	Why is security important in this project? What are our main assets that we need to secure? What are our main security concerns in this project? Are there explicit security requirements from the tender/customer?
	Analysis and training needs	Compared to previous projects, are there new types of technology, new types of security requirements, new types of threats, new types of assets? Do we plan considerable design changes in existing solutions? Do we have the necessary overview to understand the security risks in the solution? Are we aware of areas where we lack the necessary security competence?
	Security coding and testing practices	Do we have the necessary coding practices to ensure the security of our code is according to our needs? Is the current practice of code review sufficient? Do we need specific types of testing for security? What about pentesting?
Security areas	Privacy	What sensitive data is handled by the project?
	Authentication and password management	Which authentication and password mechanisms will be used here and where?
	Authorisation and role management	What is the RBAC for the front-end and back-end? What are the database user accounts and privileges? Which privileges will operations have on the databases? Which privileges others will have? Does some actions need extra authentication?
	Session management	Is there any type of session definition for a "Device?" Do we need to document this?
	Cryptography and key management	Where are we using Tokens and Keys for authentication? Which are they? Do we have a procedure that is good for managing these? Should we have changes in the way we do things today?
	Network security	Are there any requirements to be added based on the list of equipment? Do we need a better description of how the connections will be done and who will be part of this? What are the requirements for the network setup on the Server side? Who will be responsible for intrusion detection? What are the trust boundaries here? Do we need to describe back-end access and how it is secured? How is the access to the network? What will be the privileges?
	Audit logging and analysis	Which logging is needed for different reasons, such as repudiation and for detecting attacks or problems in the system? Is there any logging of actions we should do for cases of auditing in our systems?
	Attack detection	Who is responsible for doing performing attack detection? Do we depend on any other parties, e.g., where we are not the only one officially managing the network and resources? Which type of attacks will we try to detect? Will we revoke a device after it was registered? How do we assure the integrity of the data about devices? Can we revoke registration of a device? Which situations should we do that?
	Incident management	What incidents can happen? What procedures do we need to have in place for incident management? Do we need to train for any specific incidents?
	Physical security	What are the components that we need to physically protect? How will we protect these components? What is it that we are assuming that makes the device secure in the platform? Can we revoke registration of a device? Which situations should we do that? Are there recommendations that we need to give to the customers?
	Availability protection	Which types of assets are important to be available all the time? When do we need to take backup? Who is responsible of backup of what? What are the requirements to backup of data?

(continued on next page)

cases where other security activities had been done that were not on the list, these were also informed about and discussed. Then followed an open discussion about security concerns that should be noted down and addressed. After this open discussion, they opened the excel sheet where they had recorded all previously identified security concerns. This excel sheet contained a short description of each concern, in addition to information on which application it concerned, what was the status, what was the priority, and who (if any) was responsible. Then, after going through the excel sheet, they decided on a set of activities to focus on

in the next period. Throughout the meeting facilitator took notes that were shown on screen. Notes could be taken directly in a meeting memo, or in the excel sheet.

The meeting in MediumCo and SmallCo had a different structure. These meetings made use of support material that offered a set of areas to cover and questions to aid in making decisions. These were organised in a Confluence page in MediumCo and an excel sheet in SmallCo. The security areas were adapted from Firesmith (2003) for use in MediumCo, and the questions were initially developed based on the needs of the first project where





**Table 11** (continued).

Area	Supporting questions
Data security and integrity	Who will take care of the integrity of the data that we get from third parties? Do we need to worry about this? On cases of corruption of the data, how do we recover? Which types of procedures for database protections for the back end we will have? How are we planning to make sure the data on each device is correct and updated? Do we need? How are we planning to have backup in case the data gets corrupted?
Third party component analysis	Which third party components will be involved here and how this can affect security? What are the entry points from other systems? Where is data integrity most important? For what data, for what functionality, for what input?
Release notes	NA
Release notes to operations	NA
Release notes to customers	NA

these meetings were applied. Fig. 10 shows how the supporting Confluence page looked like in MediumCo. Table 11 gives an overview of all the security areas covered in the support material, in the form used by SmallCo, MediumCo and SmallCo used the same security areas but had slightly different support for the overall concerns (in MediumCo covered by the Section Background in the Confluence page, see Fig. 10). Moreover, the supporting questions had been made less company and technology-specific before bringing them to SmallCo.

In a typical meeting in MediumCo and SmallCo, they started with going through the action points from the previous meeting before the facilitator selected a few security areas to focus on in the meeting. These were then discussed, and notes were taken (visible on screen) on decisions, concerns, and open issues identified in the discussions. As part of this they identified action points. To exemplify, one meeting in MediumCo had already identified in the agenda three areas from the checklist they wanted to discuss (privacy, network security, and key management). Then in the meeting they discussed the associated questions and any already noted concerns on Confluence, and updated the documentation in Confluence on these issues. Then, due to extra time, they moved on to discuss a few more security areas as well. In the initial meeting, the main emphasis was on the open issues in the beginning of the template (e.g., what are our main concerns to security in this project?) while later meetings went more into detail on the different areas, and revisited previous decisions and discussions. The exception to this approach was when the meeting series was brought to the new team in MediumCo, and the team managed to go through the full template in one meeting.

## Appendix B. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.jss.2022.111477>.

## References

Ahmad, M.O., Marikula, J., Oivo, M., 2013. Kanban in software development: A systematic literature review. In: 2013 39th Euromicro Conference on Software Engineering and Advanced Applications. <http://dx.doi.org/10.1109/SEAA.2013.28>.

Assal, H., Chiasson, S., 2019. Think secure from the beginning: A survey with software developers. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. <http://dx.doi.org/10.1145/3290605.3300519>.

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, A.M., Rodríguez, P., Franch, X., Oivo, M., 2020. Management of quality requirements in agile and rapid software development: A systematic mapping study. *Inf. Softw. Technol.* 123, 106225. <http://dx.doi.org/10.1016/j.infsof.2019.106225>.

Bernsmed, K., Cruzes, D.S., Jaatun, M.G., Iovan, M., 2022. Adopting threat modelling in agile software development projects. *J. Syst. Softw.* 183, 111090. <http://dx.doi.org/10.1016/j.jss.2021.111090>.

Bishop, D., Rowland, P., 2019. Agile and secure software development: An unfinished story. *Issues Inf. Syst.* 20 (1).

Crawley, B., Glas, B., Jenkins, B., Cooper, C., Kefer, D., Parekh, H., Dileo, J., Ellingsworth, J., Kennedy, J., Kisserli, N., Duarte, P., Arriada, S., Kravchenko, Y., 2020. Presenting OWASP SAMM - OWASP SAMM V2.0 - Core Model Document. OWASP. <https://github.com/OWASP/samm/blob/master/Supporting%20Resources/v2.0/OWASP-SAMM-v2.0.pdf>.

Cruzes, D.S., Jaatun, M.G., Bernsmed, K., Tøndel, I.A., 2018. Challenges and experiences with applying microsoft threat modeling in agile development projects. In: 2018 25th Australasian Software Engineering Conference. ASWEC. <http://dx.doi.org/10.1109/ASWEC.2018.00023>.

Davison, R., Martinsons, M.G., Kock, N., 2004. Principles of canonical action research. *Inf. Syst. J.* 14 (1), 65–86. <http://dx.doi.org/10.1111/j.1365-2575.2004.00162.x>.

Davison, R.M., Martinsons, M.G., Ou, C.X., 2012. The roles of theory in canonical action research. *MIS Q.* 36, 763–786. <http://dx.doi.org/10.2307/41703480>.

Deterding, N.M., Waters, M.C., 2021. Flexible coding of in-depth interviews: A twenty-first-century approach. *Sociol. Methods Res.* 50 (2), 708–739. <http://dx.doi.org/10.1177/0049124118799377>.

Firesmith, D.G., 2003. Common Concepts Underlying Safety Security and Survivability Engineering. The Software Engineering Institute, Carnegie-Mellon University. [https://resources.sei.cmu.edu/asset\\_files/TechnicalNote/2003\\_004\\_001\\_14198.pdf](https://resources.sei.cmu.edu/asset_files/TechnicalNote/2003_004_001_14198.pdf).

Fitzgerald, B., Stol, K.-J., 2014. Continuous software engineering and beyond: Trends and challenges. In: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering. <http://dx.doi.org/10.1145/2593812.2593813>.

Heeager, L.T., Nielsen, P.A., 2018. A conceptual model of agile software development in a safety-critical context: A systematic literature review. *Inf. Softw. Technol.* 103, 22–39. <http://dx.doi.org/10.1016/j.infsof.2018.06.004>.

Jarżębowicz, 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. (Eds.), *Lean and Agile Software Development*, Vol. 408. LASD 2021, Springer, Cham, pp. 91–110. [http://dx.doi.org/10.1007/978-3-030-67084-9\\_6](http://dx.doi.org/10.1007/978-3-030-67084-9_6).

Kocksch, L., Korn, M., Poller, A., Wagenknecht, S., 2018. Caring for IT security: Accountabilitie, moralities, and oscillations in IT security practices. In: *Proc. ACM Hum.-Comput. Interact.* 2, CSCW. <http://dx.doi.org/10.1145/3274361>.

Kongslil, V., 2006. Towards agile security in web applications. In: OOPSLA '06: Companion to the 21st ACM SIGPLAN Symposium on Object-Oriented Programming Systems, Languages, and Applications. ACM. <http://dx.doi.org/10.1145/1176617.1176727>.

Lindström, Y., Sjøberg, D.I.K., Dingsøyr, T., Bergersen, G.R., Dybå, T., 2016. Teamwork quality and project success in software development: A survey of agile development teams. *J. Syst. Softw.* 122, 274–286. <http://dx.doi.org/10.1016/j.jss.2016.09.028>.

Maxwell, J.A., 2013. *Qualitative research design: An interactive approach*. In: *Applied Social Research Methods Series*, vol. 41. Sage publications.

McGraw, G., Allen, J.H., Mead, N., Ellison, R.J., Barnum, S., 2013. *Software Security Engineering: A Guide for Project Managers*. Carnegie Mellon University, Software Engineering Institute. <https://apps.dtic.mil/sti/pdfs/ADA617944.pdf>.

Migues, S., Erikkhan, E., Ewers, J., Nassery, K., 2021. BSIMM12 2021 Foundations Report. Synopsys. <https://www.bsimm.com/content/dam/bsimm/reports/bsimm12-foundations.pdf>.

Miles, M.B., Huberman, A.M., Saldaña, J., 2018. *Qualitative Data Analysis: A Methods Sourcebook*. Sage publications.

Moe, N.B., Dingsøyr, T., Rølland, K., 2018. To schedule or not to schedule? An investigation of meetings as an inter-team coordination mechanism in large-scale agile software development. *Int. J. Inf. Syst. Project Manag.* 6 (3), 45–59. <http://dx.doi.org/10.12821/ijispm060303>.

- Nayak, K., Marino, D., Efstathopoulos, P., Dumitras, T., 2014. Some vulnerabilities are different than others. In: International Workshop on Recent Advances in Intrusion Detection. Springer, [http://dx.doi.org/10.1007/978-3-319-11379-1\\_21](http://dx.doi.org/10.1007/978-3-319-11379-1_21).
- Newton, N., Anslow, C., Drechsler, A., 2019. Information security in agile software development projects: A critical success factor perspective. In: 27th European Conference on Information Systems. ECIS.
- Oueslati, H., Rahman, M.M., Othma, Lb, 2015. Literature review of the challenges of developing secure software using the agile approach. In: 2015 10th International Conference on Availability, Reliability and Security. <http://dx.doi.org/10.1109/ares.2015.69>.
- Palombo, H., Tabari, A.Z., Lende, D., Ligatti, J., Ou, X., 2020. An ethnographic understanding of software (in) security and a co-creation model to improve secure software development. In: Sixteenth Symposium on Usable Privacy and Security, SOUPS 2020.
- Petersen, K., Gencel, C., 2013. Worldviews, research methods, and their relationship to validity in empirical software engineering research. In: 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement. <http://dx.doi.org/10.1109/IWVSM-Mensura.2013.22>.
- Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. *Empir. Softw. Eng.* 14 (2), 131–164. <http://dx.doi.org/10.1007/s10664-008-9102-8>.
- Schwaber, K., 2004. *Agile Project Management with Scrum*. Microsoft Press.
- Stray, V., Sjøberg, D.I.K., Dybå, T., 2016. The daily stand-up meeting: A grounded theory study. *J. Syst. Softw.* 114, 101–124. <http://dx.doi.org/10.1016/j.jss.2016.01.004>.
- Strode, D.E., Huff, S.L., Hope, B., Link, S., 2012. Coordination in co-located agile software development projects. *J. Syst. Softw.* 85 (6), 1222–1238. <http://dx.doi.org/10.1016/j.jss.2012.02.017>.
- Tøndel, I.A., Cruzes, D.S., Jaatun, M.G., 2020. Achieving good enough software security: The role of objectivity. In: EASE '20: Proceedings of the Evaluation and Assessment in Software Engineering, pp. 360–365. <http://dx.doi.org/10.1145/3383219.3383267>.
- Tøndel, I.A., Cruzes, D.S., Jaatun, M.G., Rindell, K., 2019a. The security intention meeting series as a way to increase visibility of software security decisions in agile development projects. In: Proceedings of the 14th International Conference on Availability, Reliability and Security. <http://dx.doi.org/10.1145/3339252.3340337>.
- Tøndel, I.A., Cruzes, D.S., Jaatun, M.G., Sindre, G., 2022. Influencing the security prioritisation of an agile software development project. *Comput. Secur.* 118, 102744. <http://dx.doi.org/10.1016/j.cose.2022.102744>.
- Tøndel, I.A., Jaatun, M.G., 2020. Towards a conceptual framework for security requirements work in agile software development. *Int. J. Syst. Softw. Secur. Prot. (IJSSSP)* 11 (1), 33–62. <http://dx.doi.org/10.4018/IJSSSP.2020010103>.
- Tøndel, I.A., Jaatun, M.G., Cruzes, D.S., Williams, L., 2019b. Collaborative security risk estimation in agile software development. *Inf. Comput. Secur.* 26 (4), 508–535. <http://dx.doi.org/10.1108/ICS-12-2018-0138>.
- 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. In: USENIX Symposium on Usable Privacy and Security, SOUPS 2021.
- Türpe, S., Poller, A., 2017. Managing security work in scrum: Tensions and challenges. In: The International Workshop on Secure Software Engineering in DevOps and Agile Development, SeSE, pp. 34–49.
- van der Veer, R., 2019. SAMM agile guidance. <https://owasp.samm.org/guidance/agile/>.
- Villamizar, H., Kalinowski, M., Viana, M., Fernández, D.M., 2018. A systematic mapping study on security in agile requirements engineering. In: 2018 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA, IEEE, <http://dx.doi.org/10.1109/SEAA.2018.00080>.
- Weir, C., Becker, I., Blair, L., 2021. A passion for security: Intervening to help software developers. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP, <http://dx.doi.org/10.1109/ICSE-SEIP52600.2021.00011>.
- Weir, C., Becker, I., Noble, J., Blair, L., Sasse, M.A., Rashid, A., 2020a. Interventions for long-term software security: Creating a lightweight program of assurance techniques for developers. *Softw. - Pract. Exp.* 50 (3), 275–298. <http://dx.doi.org/10.1002/spe.2774>.
- Weir, C., Rashid, A., Noble, J., 2020b. Challenging software developers: Dialectic as a foundation for security assurance techniques. *J. Cybersecur.* 6 (1), <http://dx.doi.org/10.1093/cybsec/tyaa007>.
- Wieringa, R.J., 2014. *Design Science Methodology for Information Systems and Software Engineering*. Springer.
- Wieringa, R., Morali, A., 2012. Technical action research as a validation method in information systems design science. In: Design Science Research in Information Systems. In: Advances in Theory and Practice, DESRIST, vol. 212, Springer, [http://dx.doi.org/10.1007/978-3-642-29863-9\\_17](http://dx.doi.org/10.1007/978-3-642-29863-9_17).
- Williams, L., Gegick, M., Meneely, A., 2009. Protection poker: Structuring software security risk assessment and knowledge transfer. In: Engineering Secure Software and Systems, ESSoS 2009, Springer, [http://dx.doi.org/10.1007/978-3-642-00199-4\\_11](http://dx.doi.org/10.1007/978-3-642-00199-4_11).
- Williams, L., Meneely, A., Shipley, G., 2010. Protection poker: The new software security game. *IEEE Secur. Priv.* 8 (3), 14–20. <http://dx.doi.org/10.1109/msp.2010.58>.
- Wohlin, C., Aurum, A., 2015. Towards a decision-making structure for selecting a research design in empirical software engineering. *Empir. Softw. Eng.* 20 (6), 1427–1455. <http://dx.doi.org/10.1007/s10664-014-9319-7>.
- Yin, R.K., 2018. *Case Study Research and Applications*, sixth ed. Sage.

**Inger Anne Tøndel** is a Ph.D. candidate at the Department of Computer Science, Norwegian University of Science and Technology (NTNU), Trondheim. Recently, she held a position as a senior research scientist at SINTEF Digital, Trondheim, Norway, and her research interests include software security, security requirements, information security risk management, cyber insurance, and smart-grid cybersecurity. She now works as a senior advisor at the Norwegian Directorate of e-health. Tøndel received an M.Sc. in telematics from NTNU in 2004.

**Daniela Soares Cruzes** is a professor at the Department of Computer Science, NTNU, Trondheim, Norway. Her research interests are agile software development, software security, software-testing processes, empirical research methods, theory development, and synthesis of software-engineering studies. Cruzes received her Dr.Ing. in electrical and computer engineering with emphasis in empirical software engineering at the University of Campinas, Brazil, in 2007. She has two postdoctoral studies, one at the Fraunhofer Center at the University of Maryland, College Park, and one at the Norwegian University of Science and Technology, Trondheim. She is a member of committees with various highly ranked international conferences and journals.



## Supplementary material Paper H: Data collection instruments

This appendix gives an overview of the data collection instruments used in this study. The same observation template was used for all cases, while the interview guides were targeted towards the case.

### Observation template

The observation template consisted of the following parts, and combined notes taken during the meeting with reflections afterwards:

- **Event details:** date and time, type of meeting, observer(s), team, number and type of participants, meeting goal, overall agenda/structure.
- **What happened:** notes taken during the meeting by the observer – on discussions, decisions, engagement by participants, etc.
- **Reflections made after the meeting:**
  - How is what happened in this meeting related to the security intention meeting series concept? What worked well that we can learn from to improve the concept? What was challenging that we can learn from to improve the concept?
  - What kind of priorities were made, discussed, or identified related to security requirements (in a broad sense)?
  - Did any factors come up (in the meeting or in me as an observer) that could be related to security priorities, and in what way could they be related?
  - What questions did they have?
  - To what extent did everybody participate? What was the mood in the session? Were they open to try?
  - Did they make or suggest any changes to the meeting? (If so, explain.)
  - Which types of awareness do you think was created here?
  - Do you have any interesting thoughts on the session? (If so, explain.) Was anything surprising to you? For example, something that you thought would happen but didn't? Or something that you did not expect but happened?
  - What security topics were mainly discussed (threats, assets, vulnerabilities, mitigations)?
  - What worked well, what was challenging?
  - Do you think they will keep doing this? (Please justify your answer.)
  - How did the observation go? Any thoughts/opinions on how we as observers and participants may have influenced the process?

### **Interview guide MediumCo**

The interview guide used for the semi-structured interviews consisted of four topics. First, issues concerning data management was presented and discussed. Then, the meeting discussed a recently held security decision meeting. Then, we moved on to discussing what they thought would happen after they no longer received support for facilitation of the meeting. Then, we talked about security decisions and prioritisation more generally.

- 1) Purpose of the interview, data management, consent
- 2) We want to talk to you about the security decision meeting we had recently, and similar meetings:
  - a) Do you remember having meetings of this type previously?
  - b) What do you consider the goal of such a security decision meeting?
  - c) What motivates you to participate, or what reduces your motivation to participate?
  - d) What do you experience as the effect of this meeting?
  - e) What in the meeting works well?
  - f) What in the meeting can be improved to increase its usefulness?
  - g) Could we do anything other than a meeting to get the same or a better effect on software security?
- 3) Support for facilitation of this meeting stops now, what do you think about continuing this type of meetings:
  - a) If you get the responsibility, will you continue with such security decision meetings?
  - b) Who should be responsible for security decision meetings?
- 4) The meeting is about security decisions and prioritisation
  - a) How is it for you to make security priorities?
  - b) How do you think this is for other product owners with less technical knowledge?
  - c) What factors influences what focus you give to security?
  - d) How can a product owner be supported in making good security decisions? Can such a meeting play a role? If so, how?

### **Interview guide SmallCo**

The interview with SmallCo emphasised adoption. This was grounded in our research questions, but also in our experiences from MediumCo, that showed that adoption was challenging. Before the interview at SmallCo, the interviewee filled out a questionnaire based on the Technology Acceptance Model (TAM) (Davis 1989) to evaluate the meeting. The questionnaire was a modification of questionnaires used in previous studies (Dyba et al. 2004; Tøndel et al. 2018a;

Tøndel et al. 2018b). This questionnaire was not intended to be used in a quantitative analysis, but rather to be used as a basis for discussion in the interview. The interview guide for the semi-structured interview was as follows:

- 1) Purpose of the interview, data management, consent
- 2) Compatibility:
  - a) How is this way of working compatible with your work practices in this company?
  - b) What could be done to make it a better match?
  - c) Discuss related questionnaire responses.
- 3) Complexity and triability:
  - a) How difficult or easy is it for you to work with the excel sheet? What makes it easier? What makes it more difficult?
  - b) How easy or difficult is it to use the excel sheet as a basis for a security meeting? What could make it easier
  - c) How do you think it would be to start using the excel sheet and have such meetings without our help? Would it be possible? What would be most challenging?
  - d) Discuss related questionnaire responses.
- 4) Observability and relative advantage:
  - a) What is the result of using this technique?
  - b) How visible are these results?
  - c) What could be done to have more visible results?
  - d) To what extent do you find the time spent is worth it? The time in the meeting, the time to follow up, the time spent in the first project, time spent in later projects?
  - e) Discuss related questionnaire responses.
- 5) Adoption in general:
  - a) What are your thoughts on continuing this practice without our support?
  - b) What would lead you to continue using this or other techniques for security?
  - c) What leads to you stopping using the techniques?
  - d) Our last meetings has been less centred on the excel sheet than those in the beginning, do you have any thoughts on that?
  - e) Discuss related questionnaire responses.
- 6) Usefulness outside of the meeting:
  - a) Have you used the documentation created outside of the meeting? If so, how? How did it work?
  - b) Have the security priorities made in the meeting had any effect on your work?
- 7) Wind up:
  - a) Do you have any more comments on the meeting, or any recommendations for others that would consider adopting this meeting type?

## Interviews and retrospective UnaidedCo

Before officially starting the case study, we performed an evaluation with UnaidedCo to identify and evaluate their software security practices, based on BSIMM (Migues et al. 2021). This evaluation consisted of individual interviews with developers and a group session with senior in the development department where we went through all BSIMM activities and considered to what extent they were adopted. The individual interviews used the following interview guide:

- Purpose of the interview, data management
- Can you tell us about your role and what you are developing?
- Can you tell us about your software development process, including how you work with security?
- How do you identify the security needs and requirements?
- How do you consider how much time you need to spend on identifying security requirements or threats towards the system?
- Do you do any analysis of security in the architecture/design, or do you do any threat modelling?
- Do you have guidelines for how to consider security during coding?
- Do you have any practices on evaluating the security of the software you are developing?
- What do you do if a security vulnerability is discovered, and there is an incident?
- Do you consider that you have received the necessary training on software security?
- Who has the responsibility for software security, in your opinion?
- What do you think about having a security group in the department?

The retrospective happened after observation of the meetings. The brainstorming performed in the retrospective was supported by the tool Mentimeter. The following topics were covered:

- Session 1:
  - What is the goal of the security group? (brainstorm, discussion)
  - What is the effect of the security group? (brainstorm, discussion)
  - Do you have good enough security in your products? (vote on a scale, discussion)
  - What works well? (brainstorm, discussion)
  - What can be improved to increase the usefulness? (brainstorm, discussion)
  - Feedback on this retrospective session
- Session 2:
  - Researcher presenting experiences from other companies on software security, to trigger discussions on the software security work at UnaidedCo. Topics covered: security awareness, knowing that security activities leads to improved security in the software, people and responsibilities, development vs. operations, visibility of security, knowing what is good enough

- How is it to do security prioritisation in the meetings? How is it to do security prioritisation in your daily activities? (vote on a scale, discussion)
- What makes it difficult to do prioritisation related to security? (brainstorm, discussion)
- What helps you make prioritisation related to security? (brainstorm, discussion)
- What factors are used in security prioritisation? Which should be used? (ranking of some predefined factors identified during observation - e.g., time to do the task, effect, risk.)
- Presentation of the idea behind the security intention meeting series and the instantiation at MediumCo – as a basis for discussion
- Feedback on this retrospective session, and thoughts about the future

## References

- Davis FD (1989) Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*:319-340. <https://doi.org/10.2307/249008>
- Dyba T, Moe NB, Mikkelsen EM (2004) An empirical investigation on factors affecting software developer acceptance and utilization of electronic process guides. 10th International Symposium on Software Metrics, 2004. Proceedings., IEEE. <https://doi.org/10.1109/METRIC.2004.1357905>
- Migues S, Erlikhman E, Ewers J, Nassery K (2021) BSIMM12 2021 Foundations Report. Synopsys. <https://www.bsimm.com/content/dam/bsimm/reports/bsimm12-foundations.pdf>
- Tøndel IA, Jaatun MG, Cruzes D, Oyetoyan TD (2018a) Understanding challenges to adoption of the Protection Poker software security game. In: *Computer Security. SECPRE Cyber CPS 2018*, Springer, vol 11387, pp 153-172. [https://doi.org/10.1007/978-3-030-12786-2\\_10](https://doi.org/10.1007/978-3-030-12786-2_10)
- Tøndel IA, Oyetoyan TD, Jaatun MG, Cruzes D (2018b) Understanding challenges to adoption of the Microsoft Elevation of Privilege game. Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security. <https://doi.org/10.1145/3190619.3190633>

## SECONDARY PAPERS

The following papers are not included in the thesis, but are relevant to the content of the thesis in the following way:

- **Paper I ‘Understanding Challenges to Adoption of the Protection Poker Software Security Game’:** This paper presents results from the study of Protection Poker with students. It was not included in the thesis, as Paper C is an extended version of this paper.
- **Paper J ‘Understanding Challenges to Adoption of the Microsoft Elevation of Privilege Game’:** This paper presents results from the study of the Microsoft Elevation of Privilege game with students, in the same environment as the study presented in Paper I. The study of Microsoft Elevation of Privilege - a threat modelling game - was part of the initial empirical exploration. However, results from the Protection Poker game were more promising, and the Protection Poker game was also more in line with the need for security prioritisation. Still, experiences from studying the Microsoft Elevation of Privilege game were important in shaping this thesis.
- **Paper K ‘Challenges and Experiences with Applying Microsoft Threat Modeling in Agile Development Projects’:** This paper presents results on threat modelling, and identifies challenges to threat modeling, including threat modelling meetings. This was used as input in the work on the Security Intention Meeting Series (e.g., in Paper H).
- **Paper L ‘Using Situational and Narrative Analysis for Investigating the Messiness of Software Security’:** This paper introduces the concept of messiness in relation to software security, a concept that in this thesis is mentioned in the discussion of this thesis (Chapter 5). Furthermore, it argues that the messiness of software security should have

implications for analysis in studies of software security practices in companies. The analysis approach proposed in this paper was used in Paper H of this thesis.

- **Paper M ‘The Quality Triage Method: Quickly Identifying User Stories with Quality Risks’**: In the discussion (Chapter 5) it was pointed out that one important direction of future work was to consider quality aspects together. This paper represents one step in this direction, proposing a method to identify quality requirements and their interaction within an agile software development setting.
- **Paper N ‘SecureScale: Exploring Synergies between Security and Scalability in Software Development and Operation’**: This paper investigates the relation between two quality aspects, security and scalability, and how these can benefit from being considered more in relation to each other. Thus, it represents one step towards a more integrated approach to quality requirements (as called for in the discussion (Chapter 5)).