Investigating the Current State of Security in Large-Scale Agile Development

Sascha Nägele(✉), Jan-Philipp Watzelt, and Florian Matthes

Technical University of Munich, Munich, Germany
{sascha.naegele,jan-philipp.watzelt,matthes}@tum.de

Abstract. Agile methods have become the established way to successfully handle changing requirements and time-to-market pressure, even in large-scale environments. Simultaneously, security has become an increasingly important concern due to more frequent and impactful incidents, stricter regulations with growing fines, and reputational damages. Despite its importance, research on how to address security in large-scale agile development is scarce. Therefore, this paper provides an empirical investigation on tackling software product security in large-scale agile environments. Based on a literature review and preliminary interviews, we identified four essential categories that impact how to handle security: (i) the structure of the agile program, (ii) security governance, (iii) adaptions of security activities to agile processes, and (iv) tool-support and automation. We conducted semi-structured interviews with nine experts from nine companies in five industries based on these categories. We performed a content-structuring qualitative analysis to reveal recurring patterns of best practices and challenges in those categories and identify differences between organizations. Among the key findings is that the analyzed organizations introduce cross-team security-focused roles collaborating with agile teams and use automation where possible. Moreover, security governance is still driven top-down, which conflicts with team autonomy in agile settings.

Keywords: Large-scale agile · Security · Software development

1 Introduction

The use of agile methods is omnipresent. According to the most recent "State of Agile Report", agile adoption within software development teams has surged from 37% in 2020 to 86% in 2021 [11]. Agile development methods are also increasingly applied to large projects and companies with numerous software development teams working together [12]. Companies thereby aim to benefit from the advantages of these methods, such as enhanced adaptability to fast-evolving environments and accelerated time-to-market [37].

At the same time, software security is becoming an increasingly important concern due to stricter legislation and growing fines [9]. In addition, there is a growing intrinsic motivation for companies to pay more attention to security. As a global risk management survey with thousands of participating companies shows, cyberattacks, data breaches, and reputational damage are the most significant perceived risks to business success [4]. The global Covid-19 pandemic further exacerbates the complexity and growing number of cyberattacks as changing work conditions and consumer behavior further increase the dependence on Information Technology (IT) [14]. Despite the importance of software security in scaled agile environments, there are only few empirical studies, and more empirical research is needed [1,31,48].

This study contributes to the empirical evidence on how organizations tackle software security in large-scale agile development (LSAD). The primary research question we strive to answer is: **How is security approached in LSAD, and what are recurring best practices and challenges?** We provide a cross-industry overview based on literature and interviews with nine experts from nine companies in five industries. The remainder of this paper is organized as follows: Section 2 presents the theoretical background and related work. Section 3 explains the research methodology. Section 4 summarizes the results, which are discussed in Sect. 5. Section 6 presents the conclusion and outlook.

2 Background and Related Work

We follow Dikert et al. in defining LSAD, who speak of a minimum of 50 people or at least six teams [12].

One of the earlier related works is by Bartsch [45], who studied security in agile development by interviewing ten practitioners but does not explicitly address LSAD contexts yet. Relevant for our work is the more recent study by Amber et al. who identified three unique security challenges in LSAD: "(i) alignment of security objectives in a distributed setting; (ii) developing a common understanding of roles and responsibilities in security activities; and (iii) integration of low-overhead security testing tools" [48]. Our key findings discuss how our results relate to these challenges.

In addition, valuable related work includes widespread software security maturity frameworks, e.g., the Building Security in Maturity Model (BSIMM) [28] and the OWASP Software Assurance Maturity Model (SAMM) [35]. These are mainly driven by practical experience from the industry and provide a highly comprehensive insight into secure software development initiatives. Even if they do not explicitly address LSAD and describe themselves as agnostic of the development approach, many of the listed organizations working with these models fulfill the definition of LSAD. However, we base our study on a literature review to achieve unbiased research independent of these models.

In the following subsections, we present the theoretical background and related work using four categories that emerged from our literature review and can be mapped to Amber et al.'s [48] challenges. We also use these categories to structure our interviews and results.

Structure of the Agile Program. Poller et al. [38] emphasize that considering organizational structures, e.g., roles and their interaction, is vital for promoting security approaches and governing agile teams in LSAD. Alsaqaf et al. [1] found in a systematic literature review that additional roles are introduced in LSAD to address quality requirements, e.g., a security architect. The authors emphasize that further empirical research on such roles is needed. Newton et al. [33] discovered security-related communities of practices, while Rindell et al. [39] observed an internal software security group that, e.g., carries out security reviews. Steghöfer et al. and Dännart et al. note that LSAD frameworks do not provide security compliance out-of-the-box [10,46]. Moyon et al. [30] recommend further adaptions, e.g., by introducing security roles. Oyetoyan et al. [36] describe a group of security experts supporting with, e.g., adherence to security standards and organizing security audits. The proposal of Boström et al. [8] includes a team of security engineers, e.g., to support the definition of security stories and risk assessments together with product teams.

Also, publications from software companies such as SAP [42], Microsoft [27] and Google [15] show that dedicated security roles are being used in practice, although the exact range of tasks is not always explained in detail. We thereby derive that the *structure of the agile program* is vital for addressing security in LSAD.

Security Governance. Security governance can be seen as a subset of IT governance, often characterized by top-down control [17]. Despite limited empirical studies on IT governance in agile and lean environments, its importance has been recognized [47]. The literature recommends moving to agile and lean governance approaches to better align governance and agility. The term lean governance is more frequently used in industry publications such as white papers and large-scale agile frameworks [3,43]. Horlach et al. [16] found that traditional governance structures hinder autonomous agile teams in LSAD. Ambler [2] stated early on that a lean form of IT governance is required to achieve agility in software development at scale. Vejseli et al. [49] found that agile IT governance positively affects business-IT alignment and, thus, enterprise performance, similar to traditional governance. By fostering the necessary engagement of all parts of the business, agile governance helps increase business agility [23]. Agile governance focuses on enabling and motivating development teams through collaborative and supportive practices [2]. Instead of top-down control, it promotes bottom-up engagement, autonomy, and self-organization [3,24]. Because of this tension, we derive *security governance* as an essential category.

Security Activities. We understand security activities as a set of practices that directly or indirectly enhance software security. A typical example is threat modeling. It is a component of security risk analysis [25] and supports the identification of security risks and appropriate measures [44]. Other common examples are penetration testing [5] and code reviews [41]. Multiple researchers agree that incorporating security activities in agile development is feasible and necessary [31,33]. Beznosov and Kruchten [7] propose integration strategies depending on the match between security practices and agile principles. As stated by Keramati and Mirian-Hosseinabadi, security activities are integrated with agile software development based on balancing "the costs of decreased level of agility [...] and benefits from developing more secure systems" [19]. Hence, we derive the category of *security activities* for our interviews.

Tool-Support and Automation. In their case study, Barbosa and Sampaio note that the "demand to build software quickly and cost-effectively" impedes the integration of agile security approaches due to the associated cost and time effort [6]. Therefore, automating manual, work-intensive tasks is crucial to reduce the friction between security and iterative deployment practices. In recent years, the term DevSecOps matured from a buzzword to a well-established movement [32]. One of the primary goals is integrating security activities and practices into development pipelines facilitated by security automation tools [29]. Researchers emphasize automating repetitive manual tasks, like security code reviews, to ensure security while sustaining a high velocity in agile software development [18,34]. Examples of security automation include static and dynamic application security testing. Since reducing manual effort and a more frictionless integration of security activities is critical in scaled environments, we derive *tool-support and automation* as the fourth category for our interviews.

3 Research Methodology

We present the three stages of our methodological process below: study design, data collection, and analysis.

Study Design. To gain cross-case insights into our research question, we deemed an interview study the most suitable primary research method. We excluded a multiple case study because not enough cases provided multiple sources for data collection due to the topic's sensitive nature. To allow for a better aggregation and comparability of results, we roughly structured the interview with the categorization described in the background and related work. Before conducting the actual interview study, we performed four preliminary expert interviews in two organizations to discuss and evaluate the categorization. In each category, we used semi-structured questions, which allow for enough freedom in the answers and the possibility for individual adjustments during the interview [13].

In contrast to expert-focused surveys, we also considered the experts' current organizations, i.e., we did not select the experts solely based on their role, competency, and experience, but an important factor was the organization they currently work for. The organizations must fulfill the previously described definition of LSAD.

Data Collection. For the interview study data collection, experts from nine companies participated in our study.

We collected data across five industries to ensure better generalizability of results. The following sectors are represented based on the main product focus of the case company: IT and software development, software development consulting, media, insurance, and automotive. Two researchers interviewed six of the nine interviewees. Three were interviewed by one researcher. After obtaining explicit consent to record the interviews for transcription purposes, we used online video conferencing tools and recorded all interviews. On average, the respondents had about six years of experience with LSAD, with a minimum of three years and a maximum of fifteen years. The experts' roles included security leads of agile programs, security engineers and security champions, an IT (security) consultant, an IT (security) architect, and a product owner (PO). To protect the anonymity of our interviewees, we intentionally do not provide further details.

Data Analysis. There are several standardized methods for the analysis of qualitative material. We used the Kuckartz [20] model to analyze our interview study data because it offers a deductive-inductive possibility for coding classification formation. We conducted the content-structuring qualitative content analysis using the qualitative data analysis software *MAXQDA* [26]. The two researchers who performed the interviews also conducted the analysis.

4 Interview Results

In this section, we present the main findings from the data analysis of the expert interviews. We first overview our results, then summarize framework usage and challenges, followed by the findings in the four categories of our interviews. To ensure the anonymity of the participating organizations, we intentionally describe the results only in an aggregated format and not specific for each case, except for Table 1.

4.1 Overview

Table 1 contains a summary of the results. We identified and selected recurring best practices that emerged from the interview analysis. We classify and visualize them according to their usage in each organization through *harvey balls*. The table does not represent a complete summary, but we filtered our results for two main cases. First, the concepts with the highest recurrence, and second, concepts with the highest ratio of conflicting viewpoints among the experts. We thus prioritize displaying the most important findings based on these two criteria.

4.2 Frameworks and Challenges

Scaled Agile Framework Usage. In the beginning, we asked about the scaled agile frameworks used in the organizations. Two experts stated that their organizations adhere to the guidelines of a specific framework, in one case LeSS [22], in the other case SAFe [43]. A third and fourth expert described a more heterogeneous agile landscape where teams choose frameworks individually depending

Table 1. Overview of recurring best practices

	01	02	03	04	05	06	07	08	09
Integration of security activities									
Security self-assesment	●	●	◑	○	◑	●	●	●	◕
Bug bounty		◑		◑	●	○	●	○	●
Threat modeling	◑	●	●	●	◕	◑	◑	●	◕
Penetration testing	●	●	●	●	●	●	●	●	●
Security audits		●	●	●		●	●	◑	●
Security code review	◑	◑	●	○	●	●	○	●	◑
Tool-support and automation									
DevSecOps pipeline	◕	●	●	◕	●	◑	◑	●	●
Static code analysis	◑	●	●	●	◑	○	◑	●	●
Vulnerability scanning	◑		●	◕	●	●	◕	●	◑
Dependency checks	●	◑	●	◕		○	◑	○	◑
Security governance									
Bottom-up	◑	○	○	○	◑	○	◑	◔	◔
Top-down	●	●	●	●	●	●	●	●	●
Reusable components		●	●		◑	○	◕	◕	◑
Organizational structure									
Security champion	◑	●	○	○	●	○	●	◕	◑
Security engineers or architects	◕	●	◕	●	◑	●	●	●	●
Central security teams	●	●	◕	◕	●	◑	●	○	◕
Communities of practice	◕	●		◑	◕	●	●	●	●

none: ○ | rare or planned: ◔ | partial: ◑ | frequent: ◕ | complete: ●
no classification possible: *empty*.

on the requirements. Two experts stated that no "textbook framework" is being used for scaled agility. The remaining three experts indicated that their organizations built their own frameworks, including parts of established frameworks.

Security Challenges in LSAD. Initially, we also asked the participants about the main challenges related to security in their LSAD environment. However, we will only present challenges mentioned by at least three independent experts. The first challenge is the lack of personnel with sufficient experience in both security (governance) and agile software development. The scaled agile environment amplifies the problem because centralized security teams have frequent contact with agile teams due to short development cycles. Also, the expected response times of security experts to inquiries of agile teams are lower, resulting in a higher pressure on central security experts and possible frictions and delays in the development process.

The second challenge is the conflict between security governance and team autonomy when coordinating many teams. Teams should work as autonomously as possible, yet security policies and standards must be defined and managed. Scaling makes it challenging to monitor and control, as it is no longer possible to "look over the shoulders of the developers", as one expert stated.

4.3 Organizational Structure

All interviewed experts report that their organization is performing some sort of structural adaptations of their agile programs due to a higher relevance of security. Figure 1 shows a generalized summary of the results.

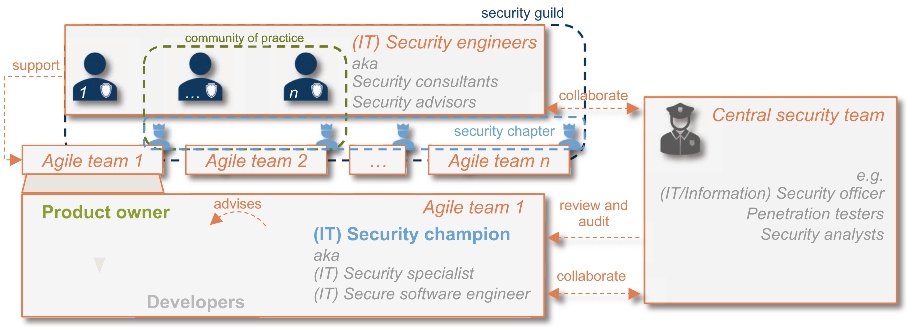

Fig. 1. Overview of organizational structure of agile programs

Centralized Security Teams. A common theme between the experts, with one exception, is that their organizations leverage existing central security teams to work with agile programs. These teams include individuals dedicated to security, e.g., penetration testers, security analysts, or information security officers. Centralized teams set overarching security quality criteria for deployments of software product increments and perform security verification. They also identify and handle compliance issues, perform risk analyses and security reviews (e.g., code review or penetration tests). Some activities such as threat modeling are performed collaboratively with individual development teams. This collaboration is beneficial for training purposes. The achieved knowledge transfer might enable agile teams to perform these activities by themselves in the future, reducing the workload of central teams. Depending on the criticality and security requirements of the software artifact, some of the analyzed organizations use central security teams for auditing and approving release-ready changes before deployments to production environments. Both threat modeling and reviews are discussed in more detail in Sect. 4.6. Members of central teams are often focused on a product area or specialized in a specific security topic or technology. As mentioned in the challenges in Sect. 4.2, central teams face scaling issues and become a bottleneck when collaborating with agile development teams.

This bottleneck motivates the introduction of new roles within the agile programs. The goal is to reduce the workload on central teams and, more importantly, increase the security capabilities and thereby the autonomy of agile teams. Based on the collected data, we distinguish between two types of security-focused roles, *team-internal* and *team-external*.

Team-Internal Roles. These agile team members continue to be developers but receive additional security training. The analyzed cases use designations such as security champion, security specialist or secure software engineer, hereafter referred to only as security champion (SC). They provide the benefit of increasing security awareness. As developers, they know their products and are also familiar with security standards and best practices. One interviewee stressed that it is essential to clarify that the whole team is still responsible for the security of their application. The SC takes the lead on security activities, serves as a fixed contact person to communicate with team-external parties, and advises other team members and the PO. Three cases do not use an SC and rely more on other measures such as automated security testing.

Team-External Roles. They are referred to as security engineers, security consultants or security advisors, hereafter referred to only as security engineer (SE). They support two to twenty teams with security expertise and are often placed between the development teams and a central security department, acting as facilitators. In some organizations, SEs conduct threat modeling workshops with development teams. In other cases, this is the responsibility of the SC, to prevent bottlenecks. SEs may also analyze laws, policies, and security best practices and ensure knowledge transfer to development teams. They specialize in a software stack or are assigned to specific development teams. Two of the analyzed cases currently have no plans to introduce a specialized security role. A solution architect is responsible instead.

Cross-Team Collaboration. Security knowledge sharing takes place through regular meetings and training. Some organizations use the concept of communities of practices. Others unite the previously described roles in so-called guilds or chapters. A difference is in the scope, frequency, and target audience for which these exchanges occur. Moreover, organizations use corporate social networks and wikis to share and document security knowledge and search for experts. However, knowledge sharing remains a challenge. Existing documentation is not always helpful due to its complexity or lack of specific details for certain combinations of platforms and software. According to one expert, providing code examples for security topics is most helpful for developers.

4.4 Security Governance

All analyzed companies mainly rely on a top-down governance approach. In most cases, centralized security governance teams create company-wide standards from applicable regulations, international standards, and best practices. The companies differ in how development teams can participate in shaping security governance. One interviewee explicitly stresses that individual teams should not influence security governance because they should prioritize the development of their product. Others grant development teams a limited say in the governing standards, allowing a partly bottom-up approach. In those cases, agile teams support shaping internal standards adjustments with sufficient justification. A promising approach for effective security governance in LSAD is providing standardized, security-focused components that teams can reuse. Interviewees mentioned that these components also simplify application security verification. Stated examples are identity and access management, validation of inputs, encryption of data, or secure communication. Challenges include outdated documentation, uncertainties about correct usage, and lack of awareness.

4.5 Tool Support and Automation

All interviewees stated that their companies use DevSecOps pipelines for their applications' build and deployment phases.

Static Application Security Testing. A common denominator is the use of static code analysis tools, which are mandatory to varying degrees. In some companies, the usage depends on project requirements and the development team's decisions. In others, it is compulsory for all applications. Depending on the criticality of the findings, teams have to meet different thresholds to deploy changes to production. False positives are a commonly reported challenge of static security testing. They are especially problematic because they may lead to developers ignoring analysis results. A particular form of static analysis is using automated dependency checks, e.g., to look for the usage of outdated open-source libraries that could introduce new vulnerabilities into the product.

Dynamic Application Security Testing. The use of dynamic application security testing is not yet as mature as static code analysis. The experts stated that there are initiatives to evaluate and establish dynamic application security testing tools. They aim to automate parts of manual penetration tests. Furthermore, the experts mentioned the use of regular vulnerability scans, e.g., to check the infrastructure of the development teams for unnecessary open ports, insecure TLS versions or cipher suites, insecure HTTP header, or other security misconfigurations. Usually, central teams provide these scanning tools. Reports are immediately made available to development teams or at regular intervals, depending on the criticality.

Metrics and Quality Gates. Automation tools that are part of a DevSecOps pipeline provide metrics, e.g., for automated deployment decisions. Those metrics might include the number of open findings, the average criticality, or a total score. For these metrics, the experts stress the importance of agreeing on thresholds for quality gates. These thresholds set the boundary of whether an application is likely to be secure enough to release to production. Due to the limited capabilities of automated tools, experts stressed not to rely exclusively on automation. As an outlook, one interviewee noted that the increasing use of machine learning might soon blur the line between the areas of security testing that can be automated and those that cannot.

4.6 Integration of Security Activities

Performing concrete activities to directly or indirectly increase the degree of security of a software product is crucial. The focus of the interviews was especially on which activities are most suitable in LSAD environments, and discussing their benefits and drawbacks. The following activities were the most discussed ones by our interviewed experts.

Code Reviews and Pair Programming. Most companies use code reviews as a form of manual intervention in developing secure applications. In two cases, pair programming is used instead as the primary quality assurance activity. A reported challenge in multiple analyzed cases is that code reviews usually deal with code quality in general (except for dedicated security code reviews), and security aspects may frequently fall short. One expert explained that they focus on automated static code analysis due to the high time consumption of code reviews. Also, other experts mentioned that code reviews are a trade-off between cost and the prospect of higher code quality. Nevertheless, one expert calls code reviews "the most pragmatic approach to developing secure software". The extent and frequency of code reviews vary. Some companies decide based on the criticality and required level of protection of the software product, while others leave it to the development teams. Especially when deploying critical code to production, organizations tend to mandate code reviews. Experts mentioned that it would be helpful to conduct security code reviews only if there was a security-relevant change. However, the crux lies in identifying those relevant changes, but automation may help in the future.

Penetration Tests and Bug Bounty Programs. All case companies regularly perform penetration tests. Both internal teams, as well as contractors, are used for this purpose. The frequency and scope vary depending on the product's criticality and size. The primary reported challenge of penetration testing is the lack of continuity because of the necessary preparation and follow-up work. Short penetration tests that only assess the changes of a smaller product increment are usually not seen as economically viable. Bug bounty programs are a valuable alternative to detect vulnerabilities continuously and provide the advantage of scaling through crowd-sourced security testers.

Security Reviews and Audits. Companies use security reviews to assess compliance with internal and external regulations. Depending on the criticality of the application, the audit frequency varies from quarterly to yearly. Reviews might include assessing system architecture or security documentation, code reviews, or penetration tests. A distinction can be made between pre-deployment and post-deployment audits. A hybrid approach is also possible, e.g., regularly using post-deployment audits and applying pre-deployment controls every few sprints, or only if a product recently failed security audits. For low-risk applications, code can be deployed before all checks have been performed. When assessing the compliance of an application with given standards, respondents pointed to the commitment to guidelines. Some are merely recommendations, while others are considered indispensable.

Threat Modeling. Because of its good fit for iterative software development, threat modeling has a high priority for the interviewees. It can be performed during the initial design phase. For continuous integration into short sprints, delta threat modeling is performed. Delta threat modeling focuses on changes of the increment. The results of threat modeling can be used to prioritize specific components for code reviews or penetration testing.

Security Self-assessments. There are two main usages for security self-assessments. First, to determine whether the product in development is compliant with policies and guidelines. Second, to determine the security relevance and criticality. Self-assessments can be an efficient tool at scale because they delegate responsibility to the teams. One interviewee stressed that the goal is to keep the number of validations by team-external stakeholders as low as possible. A benefit of self-assessments is the creation of security awareness. The concept of "comply or explain" was also mentioned. Developers may explain where they have made a conscious decision not to meet a requirement. Depending on the criticality, this might be considered during risk management. One organization deliberately avoids self-assessments because they are too time-consuming.

Security Risk Management. A recurring aspect in the interviews is the possibility to release or keep operating software with certain security risks or compliance issues, often referred to as "risk acceptances". A PO has to take responsibility for the risk and systematically document it. A SC or SE usually supports the PO to identify and report risks proactively. Furthermore, risks can also result from other activities, e.g., threat modeling, penetration testing, or security reviews. Some teams perform and document risk assessments themselves, e.g., as attributes or flags of their feature tickets or user stories.

Security Documentation. On the on hand, experts stated that extensive security documentation is often not feasible for frequent product iterations. Therefore, companies evaluate tools to automatically create documentation, e.g., risk reports generated from threat models. On the other hand, experts explained that incrementally adapting and extending existing documentation with every sprint is feasible. They suggested using existing tools to include security requirements, e.g., issue tracking software.

5 Discussion

We answer our research question by discussing the key findings and then critically describe the limitations.

5.1 Key Findings

We identified two current challenges specific to security in LSAD that at least three experts mentioned. The first challenge is the lack of qualified personnel with sufficient experience in both security (governance) and agile software development. This challenge amplifies in LSAD due to the larger number of teams. The second challenge is the conflict between security governance and team autonomy when coordinating many teams.

An essential aspect addressing the first identified challenge is the structure of the agile program. Our findings show that all analyzed cases introduce additional security roles, as recommended in the literature. We were able to identify the use of central security teams, roles within the development team, and roles outside of a team. Furthermore, we show that some organizations are not leveraging team-internal security roles, such as a SC. Nevertheless, these roles might be most effective long-term because they enable teams to perform more security activities independently, resulting in more autonomy. To support agile teams, a solid DevSecOps pipeline with static and dynamic application security testing tools is indispensable.

The second challenge fits well with our findings in the security governance category. In all of the analyzed cases, security governance is mainly driven top-down, in contrast to the recommendations from the literature. However, bottom-up approaches are beginning to establish, e.g., development team members gathering in dedicated security communities. In our opinion, leaving the definition of security standards up to individual teams results in substantial, economically unjustifiable efforts and might result in conflicts of interest. A certain level of top-down control is still necessary, e.g., to prepare for external audits. Nevertheless, agile teams should be able to influence the security governance decision-making, and top-down governance should partly shift to self-governance. The described security roles provide a good starting point for building the necessary competency in and around agile teams. This shift could be a way to find the right balance between autonomy and control, consequently bringing closer security governance and LSAD.

Finally, we would like to place our results in the context of the security challenges described by Amber et al. [48], and existing software security maturity models. Our findings regarding the structure of the agile program, security governance, and security activities provide more clarity on how to address the challenge of aligning security objectives in a distributed setting, and contribute to solving the challenge of a common understanding of roles and responsibilities. Our results in the tool-support and automation category relate to the third challenge described by Amber et al., which is "the integration of low-overhead security testing tools" [48].

We identified common patterns between our results and established software security maturity models. For example, the BSIMM [28] identifies so-called *software security groups* in the studied organizations, which are described very similarly to the observed centralized security teams in our study. Another example is the *satellite* role, whose description is largely consistent with the team-internal roles reported in our study. In this particular aspect, our study provides even more granularity by identifying and describing the team-external roles, which are even more widespread than the team-internal roles in the LSAD environments analyzed in this study. Further research on the similarities and differences between our results and software security maturity frameworks could lead to additional interesting findings.

5.2 Limitations

Even though we conducted an interview study, some of the common limitations of case studies described by Runeson and Höst [40] are also relevant for our study and help to structure our limitations. We addressed the threat of *construct validity* by clarifying any ambiguity directly during the conversation with the interviewees. To overcome the threat of *external validity*, which refers to a limited generalizability of results, we based our interviews on scientific literature and conducted the interviews in nine organizations from five industries. However, since we interviewed one expert at each company, we have only a limited picture of each organization. Companies are rarely homogeneous enough for one expert to grasp the entire situation. We countered this by designing our questions to identify overarching patterns within an organization. Additionally, we encouraged our interviewees to keep generalizability in mind. Moreover, the total number of interviewees might be considered relatively small. However, we had already reached a certain level of saturation in the sense that the data collected in the last few interviews became increasingly redundant compared to the data previously collected. To ensure *reliability*, we recorded, transcribed and coded the interviews. This analysis was documented, validated and discussed by the two researchers. Finally, typical problems arise when conducting interviews. That is why we followed the guidelines for good interviews by Kvale [21].

6 Conclusion and Future Work

Addressing security in LSAD is a significant challenge. Despite the importance, there is a paucity of research. Therefore, this paper provides insights into the research question of how security is addressed in LSAD by presenting the results of an interview study. We conducted a literature review to categorize the research topic and interview guide, resulting in four categories: agile program structure, security governance, security activities, and tool support and automation. Our interviews were conducted with nine experts from nine organizations in five industries. One of the key findings is that organizations use centralized security teams, team-internal and team-external security roles. In addition, organizations are using automation for security testing and integrating security activities such as threat modeling or code reviews. Security governance is mainly top-down, while our recommendation is to shift attention to bottom-up approaches. Our findings contribute to raising awareness of the areas to focus on when developing secure software at scale. Practitioners could leverage our results by discussing and applying the identified best practices in their organizations.

Our research could serve as the basis for further scientific investigation. The recurring best practices could be analyzed for their relative impact and effectiveness. Due to the complexity of the research topic, further research could also identify and explore other important aspects regarding security in LSAD, in addition to the four categories identified in our work. Moreover, as we suggest a shift toward more bottom-up security governance, a more in-depth study or evaluation of existing approaches could be conducted. For example, further research could focus on the impact of relevant secure software development maturity models to adapt security governance and compliance processes to agile at scale. More mature development teams may be more capable to self-govern their security posture, and their organizations may be able to afford less top-down control.

Funding. This work has been supported by the German Federal Ministry of Education and Research (BMBF) Software Campus grant 01IS17049.

References

1. Alsaqaf, W., Daneva, M., Wieringa, R.: Quality requirements in large-scale distributed agile projects – a systematic literature review. In: Grünbacher, P., Perini, A. (eds.) REFSQ 2017. LNCS, vol. 10153, pp. 219–234. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54045-0_17
2. Ambler, S.W.: Agile software development at scale. In: Meyer, B., Nawrocki, J.R., Walter, B. (eds.) CEE-SET 2007. LNCS, vol. 5082, pp. 1–12. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85279-7_1
3. Ambler, S.W., Lines, M.: Choose Your WoW. Boston (2020)
4. Aon PLC: 2019 global risk mangement survey - report — Aon (2019). https://www.aon.com/getmedia/8d5ad510-1ae5-4d2b-a3d0-e241181da882/2019-Aon-Global-Risk-Management-Survey-Report.aspx. Accessed 21 Feb 2021
5. Arkin, B., Stender, S., McGraw, G.: Software penetration testing. IEEE Secur. Priv. **3**, 84–87 (2005)
6. Barbosa, D.A., Sampaio, S.: Guide to the support for the enhancement of security measures in agile projects. In: 2015 6th Brazilian Workshop on Agile Methods (WBMA), pp. 25–31 (2015)
7. Beznosov, K., Kruchten, P.: Towards agile security assurance. In: Raskin, V. (ed.) Proceedings of the 2004 Workshop on New Security Paradigms, ACM Conferences, p. 47. ACM, New York (2004)
8. Boström, G., Wäyrynen, J., Bodén, M., Beznosov, K., Kruchten, P.: Extending XP practices to support security requirements engineering. In: Bruschi, D. (ed.) Proceedings of the 2006 International Workshop on Software Engineering for Secure Systems, ACM Conferences, p. 11. ACM, New York (2006)
9. Breaux, T.D., Anton, A.I.: Analyzing regulatory rules for privacy and security requirements. IEEE Trans. Software Eng. **34**, 5–20 (2008)
10. Dännart, S., Moyón, F., Beckers, K.: An assessment model for continuous security compliance in large scale agile environments: exploratory paper. In: Advanced Information Systems Engineering, pp. 529–544 (2019)
11. Digital.ai: 15th annual state of agile report (2021). https://digital.ai/resource-center/analyst-reports/state-of-agile-report. Accessed 21 Feb 2021

12. Dikert, K., Paasivaara, M., Lassenius, C.: Challenges and success factors for large-scale agile transformations: a systematic literature review. J. Syst. Softw. **119**, 87–108 (2016)
13. Döring, N., Bortz, J.: Forschungsmethoden und Evaluation in den Sozial- und Humanwissenschaften. S, Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-642-41089-5
14. E. U. A. for Cyber Security.: Enisa threat landscape 2020 (2020). https://www.enisa.europa.eu/topics/threat-risk-management/threats-and-trends. Accessed 21 Feb 2021
15. Google: Google security whitepaper - google cloud (2019). https://cloud.google.com/docs/security/overview/whitepaper. Accessed 21 Feb 2021
16. Horlach, B., Böhmann, T., Schirmer, I., Drews, P.: It governance in scaling agile frameworks. In: Tagungsband Multikonferenz Wirtschaftsinformatik 2018 (2018)
17. IT Governance Institute: Information Security Governance: Guidance for Boards of Directors and Executive Management. IT Pro, IT Governance Institute (2006)
18. Johnson, B., Song, Y., Murphy-Hill, E., Bowdidge, R.: Why don't software developers use static analysis tools to find bugs? In: Proceedings of the 2013 International Conference on Software Engineering, ICSE 2013, pp. 672–681. IEEE Press (2013)
19. Keramati, H., Mirian-Hosseinabadi, S.: Integrating software development security activities with agile methodologies. In: 2008 IEEE/ACS International Conference on Computer Systems and Applications, pp. 749–754 (2008)
20. Kuckartz, U.: Qualitative Inhaltsanalyse: Methoden, Praxis, Computerunterstützung. Grundlagentexte Methoden, Beltz Juventa, Weinheim and Basel, 4. auflage edn. (2018)
21. Kvale, S.: Doing Interviews, The Sage Qualitative Research Kit. Flick, U. (ed.), vol. Pt. 2. SAGE, Los Angeles (2007)
22. LeSS: Overview - large scale scrum (less) (2022). https://less.works/. Accessed 21 Feb 2021
23. Luna, A., Kruchten, P., E. Pedrosa, M.L.D., Almeida Neto, H.R., Moura, H.P.M.: State of the art of agile governance: a systematic review. Int. J. Comput. Sci. Inf. Technol. **6**, 121–141 (2014)
24. Luna, A., Kruchten, P., Riccio, E., Moura, H.: Foundations for an agile governance manifesto: a bridge for business agility. In: 13th International Conference on Management of Technology and Information Systems, São Paulo, SP, Brazil (2016)
25. Ma, Z., Schmittner, C.: Threat modeling for automotive security analysis. Security Technology 2016, pp. 333–339 (2016)
26. MAXQDA: Maxqda — all-in-one qualitative & mixed methods data analysis tool (2022). https://www.maxqda.com/. Accessed 21 Feb 2021
27. Microsoft IT: Security for modern engineering: information security & risk management (2016). https://www.microsoft.com/en-us/download/details.aspx?id=54092. Accessed 21 Feb 2021
28. Migues, S., Erlikhman, E., Ewers, J., Nassery, K.: Building Security in Maturity Model (BSIMM) Report - Version 12 (2021). https://www.bsimm.com/
29. Mohan, V., Othmane, L.B.: SecDevOps: is it a marketing buzzword? - Mapping research on security in DevOps. In: 2016 11th International Conference on Availability, Reliability and Security (ARES), pp. 542–547. IEEE (2016)
30. Moyón, F., Méndez Fernández, D., Beckers, K., Klepper, S.: How to integrate security compliance requirements with agile software engineering at scale? In: Product-Focused Software Process Improvement, pp. 69–87 (2020)

31. Moyón, F., Almeida, P., Riofrío, D., Méndez Fernández, D., Kalinowski, M.: Security compliance in agile software development: a systematic mapping study. In: 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 413–420 (2020)
32. Myrbakken, H., Colomo-Palacios, R.: DevSecOps: a multivocal literature review. In: Mas, A., Mesquida, A., O'Connor, R.V., Rout, T., Dorling, A. (eds.) SPICE 2017. CCIS, vol. 770, pp. 17–29. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67383-7_2
33. Newton, N., Anslow, C., Drechsler, A.: Information security in agile software development projects: a critical success factor perspective. In: Proceedings of the 27th European Conference on Information Systems (ECIS) (2019)
34. Nguyen Quang Do, L., Wright, J., Karim, A.: Why do software developers use static analysis tools? A user-centered study of developer needs and motivations. IEEE Trans. Softw. Eng. **48**, 835–847 (2020)
35. OWASP Foundation: OWASP Software Assurance Maturity Model - Version 2.0 (2020). https://owaspsamm.org/model/
36. Oyetoyan, T.D., Cruzes, D.S., Jaatun, M.G.: An empirical study on the relationship between software security skills, usage and training needs in agile settings. In: 2016 11th International Conference on Availability, Reliability and Security (ARES), pp. 548–555 (2016)
37. Petersen, K., Wohlin, C.: The effect of moving from a plan-driven to an incremental software development approach with agile practices. Empir. Softw. Eng. **15**, 654–693 (2010)
38. Poller, A., Kocksch, L., Türpe, S., Epp, F.A., Kinder-Kurlanda, K.: Can security become a routine? In: Lee, C.P. (ed.) Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, pp. 2489–2503, ACM Digital Library. ACM, New York (2017)
39. Rindell, K., Ruohonen, J., Hyrynsalmi, S.: Surveying secure software development practices in Finland. In: Proceedings of the 13th International Conference on Availability, Reliability and Security, ACM Other Conferences, pp. 1–7. ACM, New York (2018)
40. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. **14**, 131–164 (2009)
41. Sadowski, C., Söderberg, E., Church, L., Sipko, M., Bacchelli, A.: Modern code review. In: Paulisch, F., Bosch, J. (eds.) Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice, pp. 181–190. ACM, New York (2018)
42. SAP: The secure software development lifecycle at sap (2020). https://www.sap.com/documents/2016/03/a248a699-627c-0010-82c7-eda71af511fa.html. Accessed 21 Feb 2021
43. Scaled Agile Framework: Safe 5.0 framework (2022). https://www.scaledagileframework.com/. Accessed 21 Feb 2021
44. Shostack, A.: Threat Modeling: Designing for Security. Wiley, Indianapolis (2014)
45. Bartsch, S.: Practitioners' perspectives on security in agile development. In: 2011 Sixth International Conference on Availability, Reliability and Security (ARES), pp. 479–484 (2011)
46. Steghöfer, J.-P., Knauss, E., Horkoff, J., Wohlrab, R.: Challenges of scaled agile for safety-critical systems. In: Franch, X., Männistö, T., Martínez-Fernández, S. (eds.) PROFES 2019. LNCS, vol. 11915, pp. 350–366. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35333-9_26

47. Sulejman, V.: It governance and its agile dimensions: exploratory research in the banking sector. In: Proceedings of the 52nd Hawaii International Conference on System Sciences 2019 (2018)
48. van der Heijden, A., Broasca, C., Serebrenik, A.: An empirical perspective on security challenges in large-scale agile software development. In: Oivo, M. (ed.) Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 1–4. ACM, New York (2018)
49. Vejseli, S., Rossmann, A., Connolly, T.: Agility matters! Agile mechanisms in it governance and their impact on firm performance. In: Bui, T. (ed.) Proceedings of the 53rd Hawaii International Conference on System Sciences. Hawaii International Conference on System Sciences (2020)