Assessing Team Security Maturity in Large-Scale Agile Development

Sascha Nägele
Technical University of Munich
sascha.naegele@tum.de

Jan-Philipp Watzelt
Technical University of Munich
jan-philipp.watzelt@tum.de

Florian Matthes
Technical University of Munich
matthes@tum.de

Abstract

Organizations struggle to balance agile team autonomy and strict security governance in large-scale agile development environments. In particular, conventional top-down IT governance mechanisms often conflict with the desired autonomy of decentralized agile teams. Our research presents a novel approach to resolve the tension between security governance and development agility: a criteria-based security maturity assessment that enables greater autonomy for mature agile teams. Leveraging design science research, a literature review, and an interview study, we introduce two key contributions: a criteria catalog for evaluating a team's capabilities and a team security maturity model. Our expert evaluation confirms their value for systematically assessing the teams' capabilities to deliver secure and compliant applications, allowing organizations to grant more autonomy to mature teams and prioritize supporting lower-maturity teams. Future work could go beyond expert interviews and implement and evaluate the team security maturity model through a case study or experiments.

Keywords: Large-scale agile development, team maturity, security, governance, compliance

1. Introduction

As organizations increasingly adopt scaled agile methods to achieve benefits such as faster time-to-market (Uludağ et al., 2021), they simultaneously face mounting security challenges due to escalating threats and more stringent legislation (Rindell et al., 2021). This dynamic creates a unique set of challenges within large-scale agile development (LSAD) environments (van der Heijden et al., 2018), fueling a tension between the necessity for centralized security governance and the desire for autonomy among agile development teams (Horlach et al., 2018; Nägele et al., 2022).

A potential solution to alleviate this tension is to evaluate the team maturity and capability, and grant autonomy accordingly (Poth et al., 2021). More mature development teams may be more capable of self-governing their security posture, requiring less top-down control (Nägele et al., 2022).

However, concrete guidance on how to assess the security maturity of agile development teams is scarce. None of the maturity models identified in the literature focus on the security capability of agile teams, which is why we also considered models from non-academic sources. These existing maturity models either focus on organization-wide assessments and therefore are not usable for maturity assessments of individual teams, or capture team security maturity in the form of activity checklists rather than measuring the maturity of each activity. Furthermore, they rely solely on single assessment types like self-assessments and do not specifically address the autonomy-control friction in LSAD.

To fill this gap, our research question (RQ) is:

RQ: How can a maturity model be designed and implemented to assess a team's capability to develop secure and security-compliant applications in LSAD?

We employ a design science research (DSR) methodology to create a team security maturity model that addresses the identified gap in academic literature and the shortcomings of existing models. To increase the rigor and relevance of our artifact, we conducted a systematic literature review (SLR) and an interview study within our DSR approach.

Our results provide practical guidance on how to assess the security capability of agile teams in LSAD

URI: https://hdl.handle.net/10125/107257
978-0-9981331-7-1

settings and balance the autonomy control tension.

The paper is structured as follows. Sections 2 and 3 cover background and related work. Section 4 explains our research method. Section 5 introduces our ten criteria for team security maturity that build the basis for our maturity model, presented in Section 6. Section 7 summarizes the results of our evaluation, and Sections 8 and 9 discuss and conclude our findings.

2. Theoretical background

According to Dikert et al. (2016), *LSAD* environments comprise at least 50 people or six teams. We selected this definition for our study due to its systematic derivation from a mapping study and its use among other LSAD researchers (Uludağ et al., 2021).

In our study, *security* denotes a subset of *information security*, which aims to "ensure business continuity and minimize business damage by preventing and minimizing the impact of security incidents" (von Solms, 1998, p. 224). This involves safeguarding the availability, integrity, and confidentiality of information and systems (Bell et al., 2017). Given our research emphasis on development teams, our focus within information security is on secure software development and application security, targeting risk minimization through technical and organizational measures during software application development and operation (Bell et al., 2017). *Security compliance* refers to adherence to security requirements (Julisch, 2008), which may stem from two sources: external, such as regulatory bodies or industry standards, and internal, from policies and guidelines.

To assess the capability of teams to fulfill such requirements, we propose using a maturity model. The most frequently cited type of maturity model in literature is the Capability Maturity Model Integration, a derivative of the Capability Maturity Model (CMM) (Wendler, 2012). However, so-called *maturity grids* represent an important alternative to these models (Maier et al., 2012). They can be used both as an assessment and as an improvement tool and differ from CMM-based models in the aspects of work orientation, mode of assessment, and intention (Maier et al., 2012).

3. Related work

We identified six existing maturity models relevant to our objective of assessing team security maturity. We excluded models aiming to assess the agility of development teams from our research scope, since they do not provide actionable guidance for our research goal to assess security capabilities. Only two of the six models originate from academic literature, but they both do not focus on security. Due to this scarcity of applicable models, we included non-academic sources as explained in Section 4, which resulted in four additional relevant models. We categorize them into two types: organization-level and team-level models.

Organization-level security maturity models, namely the Software Assurance Maturity Model (SAMM) (OWASP Foundation, 2022) and the Building Security in Maturity Model (BSIMM) (Synopsys, 2021) are primarily oriented towards whole-organization assessment, deviating from our study's focus on team assessment. However, we deem these models relevant to our study because they offer guidance transferable to the context of individual teams and provide instructions for integrating maturity models into overarching organizational processes.

Poth et al. (2021) introduce a team maturity model designed to assess team performance, adhering to the principle that higher maturity leads to more autonomy. While not primarily focused on security, the model incorporates security as one of its pillars within its fundamental structure of pillars, domains, and topics.

Pagel's (2020) DevSecOps Maturity Model (DSOMM) provides security measures and prioritization when deploying DevOps teams. The model utilizes dimensions, sub-dimensions, and maturity levels. Dimensions represent categories, such as "build and deploy", and sub-dimensions further specify those dimensions, e.g., "patch management". DSOMM describes sets of activities required to achieve a certain maturity, whereas our model proposes examining maturity for each capability.

The Security Belts model (AppSecure.nrw, 2021), inspired in part by DSOMM and SAMM, similarly structures security capabilities and assesses team maturity. As in our model, its primary focus is on assessing the maturity level of teams for secure software development. The structural approach of this model assigns maturity levels based on performed activities, akin to a checklist. In contrast, our proposition involves investigating maturity levels within each topic.

Finally, Britto et al. (2016) describe in a case study how Ericsson uses team maturity levels to adjust the responsibility of distributed development teams, among other findings. Although not directly linked to security, this study exemplifies practical applications of team maturity in large-scale environments.

To summarize, our proposed model distinguishes itself by employing a maturity grid system with textual descriptions for each topic's maturity level. Moreover, our model incorporates a multi-source approach, combining self-assessments, external assessments, and

automated metrics to bolster result validity. Uniquely, our model's creation and evaluation occur specifically within the context of security and LSAD, a feature not found in the other models. Finally, we close a gap in academic literature, since all the presented models focused on the security maturity of development teams stem from non-academic sources.

4. Research method

To address our RQ, we utilized the DSR process of Peffers et al. (2007) because it enabled us to systematically create and evaluate a solution artifact, the TSMM, to address an identified problem, in our case, the autonomy control tension in security and LSAD. Rigor was ensured by conducting an SLR while relevance was obtained through interviews and workshops with industry experts. We chose these methods because they are typical examples of suitable methods to create maturity models in the context of IT (Becker et al., 2009).

Further details of these methods are provided below, and additional information can be found in our research protocol and supplementary material (Nägele et al., 2023).

4.1. Systematic literature review

Our SLR aimed to identify influencing factors for developing secure and security-compliant applications in LSAD, as well as existing team security maturity models. We divided our SLR into four phases. These phases are derived from Webster and Watson's (2002) suggestions for structuring a literature review and described in the following.

Phase 1 - Foundation: We chose our databases - ACM, Web of Science, Science Direct, IEEE, Google Scholar, and Scopus - based on initial results and the selection of other LSAD researchers (Dikert et al., 2016). Subsequently, we formulated and iteratively refined our search string, resulting in: *('large-scale agile' OR 'agile at scale') AND 'software' AND ('team' OR 'teamwork') AND ('maturity' OR 'assessment' OR 'self-assessment' OR 'capability' OR 'quality' OR 'security').*

Phase 2 - Synthesis and analysis: In the second phase, we applied our search string, aggregated the results, and initiated the analysis process by eliminating duplicates and screening titles for relevance. For Google Scholar, we used the additional "exclude citations" and "review article" filters to reduce the number of results. From the resultant 138 publications, we evaluated abstracts and applied exclusion criteria prior to full-text review. We excluded publications predating the 2001 Agile Manifesto, non-English full texts, and publications outside of journals or conferences. This resulted in 51 papers, of which we deemed ten relevant based on full-text analysis.

Phase 3 - Extension: We enriched our literature set via backward and forward searches. Backward snowballing, applied to the background and related work sections of the most relevant publications, identified 28 new candidate articles. Forward snowballing involved using Google Scholar to track all articles citing the four publications we deemed most relevant, yielding 44 more candidate publications. Following the procedure of the second phase, we merged and screened the new titles from both methods. Since we could not find any maturity models that assess the security competence of development teams, and only two maturity models that are at least partially relevant to our research objective, we decided to include non-academic sources. This led to four additional models for our related work discussed in Section 3. We consider this reasonable in light of the application-oriented nature of our research. Overall, the third phase led to the inclusion of 12 additional publications.

Phase 4 - Evaluation: We identified and selected recurring best practices on how to evaluate the security capability of development teams from our 22 identified publications and documented and categorized the results in a concept matrix, as proposed by Webster and Watson (2002). The concept matrix is the basis for our written analysis and presentation of our results in Section 5.

4.2. Interviews and workshops

We conducted expert interviews to demonstrate and evaluate our artifacts to ensure practical relevance and applicability. We used the ACM standard (ACM SIGSOFT, 2023) for qualitative surveys to guide our interview study. In the following, we will briefly explain the study design and data collection and analysis.

Study design: We created and used a semi-structured questionnaire because it provides stringent interview guidance while allowing enough freedom in the answers and the possibility for individual adjustments during the interview, e.g., based on the experts' expertise (Döring et al., 2016). The experts were acquired through LinkedIn and our research network. The interviewee roles included (information) security consultants and architects, secure development experts, a senior secure development researcher, and a software quality and governance expert. Represented industries are automotive, finance, engineering, and media.

Data collection: We conducted synchronous interviews with one participant at a time using online videoconferencing tools. In total, we conducted 12 interviews. The average interview duration was 50

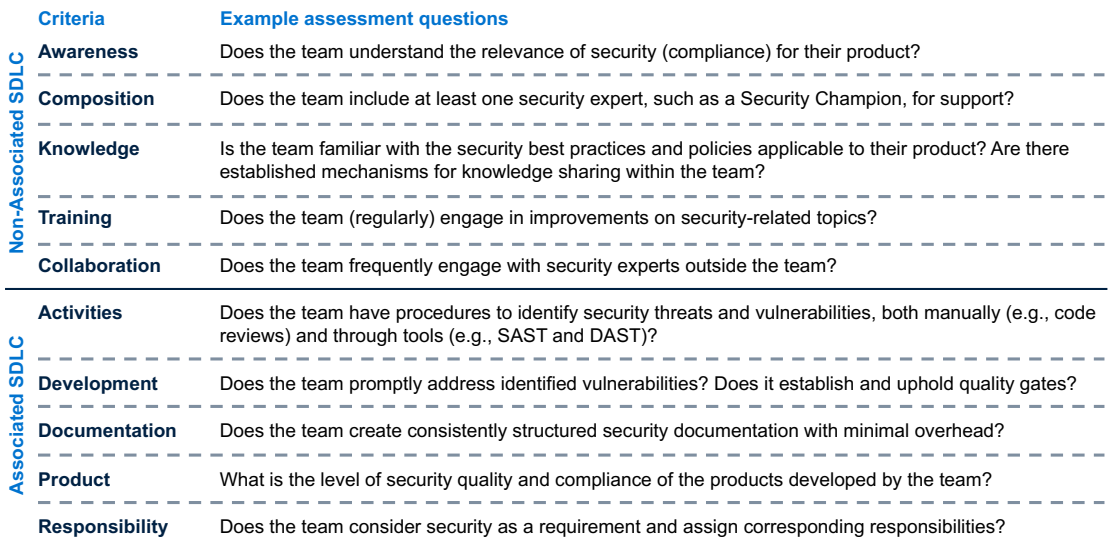

Figure 1. Overview of the team security maturity criteria

minutes. The average security experience of our interviewees was six years, and seven years in scaling agile. With the consent of the participants, we recorded the sessions and transcribed them.

Data analysis: We used the approach by Kuckartz (2016) to analyze our interview study data because it offers a deductive-inductive classification of interview statements. We conducted the content structuring analysis of our interview transcripts using the qualitative data analysis software *MAXQDA*.

As a supplementary method, we organized two workshops with two secure development experts active in LSAD to facilitate in-depth discussions of our solution artifacts. These workshops, each spanning three hours, offered the possibility for a more comprehensive discourse compared to the interviews and coined the selection of the security maturity criteria and related recommendations, as well as the structure and content of the TSMM.

5. Team security maturity criteria

We propose ten criteria for assessing team security maturity, laying the groundwork for our maturity model. Inspired by Bishop and Rowland's (2019) literature review structure on agility and security, we categorize the criteria into two groups: non-associated and associated with the software development lifecycle (SDLC) phases, each containing five criteria. Both categories are crucial for a mature team, which we characterize not only by security competencies during development, but also by overarching criteria like security awareness, team composition, and collaboration with other roles, which is especially important in the context of LSAD.

Figure 1 provides an overview of the criteria and exemplary measurement questions. In the following, we explain the criteria by presenting the relevant theory and our recommendations and suggestions, influenced by our expert interviews.

5.1. Criteria non-associated with the SDLC

Awareness: Teams often view security as nonessential in software development (van der Heijden et al., 2018). However, understanding its importance fosters responsible handling of security requirements and boosts motivation (Bodin & Golberg, 2021). Although interconnected, we consider awareness and security knowledge separate criteria to emphasize their importance. The experiences of our interviewees reveal crucial differences in the prioritization of security when teams comprehend specific risks linked to their product rather than merely fulfilling obligations. This understanding enables careful selection of suitable security measures, potentially reducing long-term efforts. To assess maturity, we suggest inquiring whether teams understand the product-specific security relevance of their system.

Composition: A team's composition, such as the inclusion of a security champion (SC), significantly influences its capacity to develop secure applications (Jaatun & Soares Cruzes, 2021). The SC, typically a software engineer specialized in security and often instrumental in raising awareness and quality, guides the team with security-related tasks like risk analysis or security code reviews. Continuous training and

experience sharing are crucial for the SC (Pagel, 2020), who often collaborates with other teams in security communities (Nägele et al., 2022).

We suggest evaluating teams based on the presence of at least one security-knowledgeable developer and their commitment to continuous learning and ability to share knowledge with the team.

Knowledge: Developing high-quality products necessitates knowledge of relevant best practices, standards, and their application (Poth, Kottke, & Riel, 2021). Mature teams incorporate security early by understanding design principles and adhering to industry or internal company standards (AppSecure.nrw, 2021). Thus, knowledge sharing within the team is essential and can be promoted by dedicated roles such as an SC (Bodin & Golberg, 2021), as previously described. An evaluation should ascertain whether the team comprehends the security best practices germane to their product and if efficient knowledge-sharing mechanisms are established.

Training: Enhancing security competency of team members reduces security risks (Bartsch, 2011; van der Heijden et al., 2018) and is essential for organizations (Synopsys, 2021). On-demand self-learning resources are particularly advantageous in LSAD environments (Poth et al., 2020). Hands-on security training, specifically tailored for developers, offers substantial value, for instance, by illustrating common software vulnerabilities (Bodin & Golberg, 2021). Therefore, measurement should focus on whether the team regularly improves its capability on security-related subjects.

Collaboration: Beyond internal teamwork, collaboration with external stakeholders is crucial for security in LSAD (Nägele et al., 2022). Agile teams may lack expertise to address security issues independently. In practice, security engineers (SEs) or (security) architects support on request (Britto et al., 2016), and agile teams collaborate with information security officers (ISOs) to discuss risks, potential countermeasures as well as their implementation (van der Heijden et al., 2018). Thus, the team maturity assessment should incorporate factors such as the quality and frequency of interactions with external stakeholders like SEs or ISOs.

5.2. Criteria associated with the SDLC

Activities: Conducting security activities is indispensable throughout the SDLC for enhancing software security. They bolster a software product's security posture and augment security knowledge and awareness (Nägele et al., 2022). Hence, the selection, quality, and frequency of security activities performed by a development team can serve as an indicator of their maturity level. The more a team refines its proficiency in individual security activities, the greater its overall security posture is enhanced. The simplest maturity assessment based on this criterion may examine whether a team routinely engages in suitable activities. These could include, e.g., threat modeling, penetration tests, the application of static (SAST) and dynamic application security testing (DAST) tools during development, or security code reviews prior to deployments. More advanced assessments could also examine the quality of those activities.

Development: Mature teams excel at preemptively preventing or fixing vulnerabilities during development (Pagel, 2020). This early remediation is beneficial because fixing defects during testing or maintenance is significantly more expensive than in the development phase (Dawson et al., 2010). An important mechanism to detect security issues as early as possible is the usage of quality gates, which could be defined as a minimum requirement to achieve higher maturity (AppSecure.nrw, 2021). Our model recommends encouraging automated capabilities during the team's journey to increase their security maturity. Instead of manual reviews, automated quality gates, e.g., through SAST and DAST tools, should be used whenever the criticality of the release allows it. Organizations could also evaluate if their teams learn from past vulnerabilities and incorporate their detection into the development process.

Documentation: Writing sufficient documentation without impeding agility presents a considerable challenge in agile development (Beznosov & Kruchten, 2004). Proper documentation is crucial for security compliance and enhances transparency, fosters understanding, and aids in maintaining and further developing software (Alsaqaf et al., 2017). In addition, specific security activities, such as threat modeling, necessitate a certain level of documentation, for example, architectural diagrams. Given the significance of documentation, we incorporate it as a criterion for team maturity assessment. However, we advise evaluating not merely the documentation quality but also the level of effort and its integration into iterative workflows. Security documentation should maintain a delicate balance between a sufficient level of detail, consistent structure, and low overhead.

Product: The quality of the outcomes produced by teams can serve as a measure of maturity. Audits conducted by external organizations can provide an objective evaluation of a software product (Bartsch, 2011). Such audits may also indirectly motivate developers to write secure code to avoid potential embarrassment (Bodin & Golberg, 2021). Examples include penetration testing, bug bounty programs, or security code and architecture reviews (Synopsys, 2021).

Alternatively, internal resources could be deployed for the same purpose. Product assessments offer the benefit of objectively determining whether the team's performance aligns with the organization's quality benchmarks, facilitating comparative analysis across teams (Bartsch, 2011). We propose that the quality of a team's product, as determined by non-team members through reviews, be used as an evaluation criterion for their security maturity. We further recommend evaluating the team's response to feedback, such as how identified security risks or vulnerabilities are addressed.

Responsibility: The responsibility for security in development teams is often unclear (van der Heijden et al., 2018), making it crucial yet challenging to precisely articulate security requirements, integrate them into agile workflows, and distribute responsibilities. Issues may arise when central security teams issue vague or overly prescriptive requirements, worsened by interdependencies and shared accountability in scaled environments, or conflicting requirements from various SEs or ISOs (Alsaqaf et al., 2017). Hence, it is essential to evaluate if a team regularly identifies security requirements, integrates them into agile methods, and clearly assigns responsibilities.

6. Team security maturity model (TSMM)

The TSMM is an exemplary model to determine the maturity of a development team in developing secure and security-compliant applications, based on the criteria previously presented in Section 5.

In the following, we first outline the structure of our model and then explain its composition. More details can be found in our supplementary material (Nägele et al., 2023).

6.1. Overall structure

To diagnose and enhance the state of development teams with minimal complexity, we decided to construct a maturity grid, aligning with the inherent objective of such models (Maier et al., 2012). Its assessment mode, featuring textual description to determine maturity levels, lends practicality to our model by enabling teams to understand the maturity levels for each topic more easily, thereby allowing better self-classification of their maturity. We chose four maturity levels for each topic to avoid an "escape category" that may emerge from an odd number of response options because it is perceived as a mean (Porst, 2011).

We propose three data sources to optimize determining the maturity score: Self-assessments by development teams, assessments by roles external to the team, and systems from which data can be extracted (semi-)automatically.

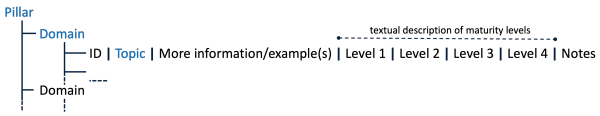

Figure 2. Outline of the TSMM structure

Following Poth et al. (2021), our TSMM is structured into pillars, domains, and topics, as shown in Figure 2. The TSMM contains three *pillars*, corresponding to the data sources, with each pillar subdivided into *domains* for grouping similar content, and further divided into *topics* featuring precise statements for classifying maturity. We implemented the TSMM in two ways: In Excel, as a simple and universally applicable solution, and as a web application prototype.

6.2. Self-assessment

We propose that teams carry out the bulk of the TSMM assessment themselves in a decentralized manner to bolster ownership and distribute effort. Our aim is to concentrate on pivotal elements and streamline the self-assessment process to enhance acceptance. Subsequently, we briefly delineate each self-assessable domain.

Table 1. Excerpt of the TSMM knowledge domain

ID	Topic
K1	We understand the significance of security in the context of our product.
K2	We are aware of and adhere to internal and external standards relevant to our product.
K3	We facilitate knowledge-sharing sessions among team members.
K4	We identify and plan to rectify security knowledge gaps.
K5	We are aware of and use standardized components for secure development.

Knowledge: This domain primarily evaluates a team's security (compliance) knowledge. We present an exemplary excerpt of this domain from the TSMM in Table 1 for illustration purposes. Topic K1 encourages a profound understanding of the significance of security in relation to their product, which is designed to instill intrinsic motivation. Our interview study disclosed that many experts witnessed security standards and guidelines being viewed as superfluous burdens due to a lack of comprehensive understanding of their core purpose. Furthermore, we strive to nurture the exchange of security knowledge within the team through topic K3. This aims

to mitigate risks associated with knowledge siloing, such as potential loss due to sickness or departure.

Activities: The effective execution of security activities enhances product security and compliance. While the TSMM does not prescribe activities, it encourages teams to make informed decisions to choose activities best suited to their needs.

Documentation: Maintaining adequate documentation is critical for (security) compliance and enables certain security activities such as risk analysis. Thus, the TSMM topics encourage teams to create, maintain, and review security documentation while emphasizing the need for minimal overhead.

Build and deployment: This domain focuses on pre-release security, integrating the criteria *activities*, *development*, and *responsibility*. Emphasizing quality gates to uphold release security, we underscore regular component maintenance, patching, and backup creation. In addition, the topics encourage automation and low-effort security testing, urging a reproducible build process with integrated security measures. Maturity levels range from non-existent to a robust DevSecOps pipeline. In addition, a topic assesses the consideration of security throughout the SDLC for early vulnerability detection and remediation.

Organization: Mainly derived from the criteria *composition* and *collaboration*, the topics of the organization domain encourage teams that seek greater autonomy to take on defined security responsibilities. At least one developer should possess advanced security knowledge and guide the team, such as by serving as an SC or similar role. In addition, teams are encouraged to identify and consult security experts outside of the team, particularly in the context of LSAD. The model also addresses the need for defined procedures to resolve conflicting security requirements between teams and for handovers of security responsibilities, a key insight from a workshop.

6.3. External assessment

The external assessment pillar includes the two domains audit and culture and facilitates evaluation by individuals external to the team, aiming to validate and expand the self-assessments. Typical roles in an LSAD environment that could conduct such an assessment are SEs who collaborate with multiple agile teams.

Audit: The audit domain mainly integrates the *product* criterion by assessing the security maturity of the produced software products of a team. For this topic, it is constructive to use a security expert who knows the required standards and regulations that apply to the product to verify. In addition, the external reviewers could also assess the development processes, security automation, and quality gates.

Culture: The TSMM envisions external reviews not solely for technical maturity. External assessments might also be beneficial to evaluate the criteria of *training*, *composition*, and *collaboration*. The derived topics encourage teams to cultivate a culture of continuous improvement to achieve enhanced product quality. An external perspective could offer valuable insights into improvement paths. Regularly collaborating security experts may be utilized to assess the team's security maturity, drawing from their interactions with the team. Furthermore, the maturity calculations also consider participation in security communities.

6.4. Automated assessment

The automated pillar uses data from security tools, necessitating customization to each organization. Organizations must identify data sources, develop strategies for data extraction, transformation, and delivery, and select and evaluate the compatibility of performance metrics with the TSMM's four customizable maturity levels. A web application implementation of the TSMM offers flexible data integration options, while an Excel implementation, though more limited, provides options like SQL connectors. Besides the primary testing domain, the TSMM also features an auxiliary domain to outline automation opportunities.

Testing: The testing domain is based on the *activities* and *product* criteria. We propose that SAST and DAST tools are used as data sources, as well as results of manual penetration tests or bug bounty programs. We have formulated corresponding topics and suggest using suitable performance indicators, such as the amount and criticality of security findings, response times to rectify findings (aggregated by criticality), and amount and criticality of security incidents.

Auxiliary metrics: Automated metrics could extend beyond security testing tools and consider any data potentially indicative of a team's maturity level. Our topics encompass examples like the inclusion of security requirements or risk-labeled tickets in software project management tools, and the team's engagement and progress in security training tools. However, the chosen metrics should be non-intrusive, and the TSMM encourages teams to actively participate in selecting metrics and defining maturity levels, e.g., through decentralized security communities.

7. Evaluation

Generally, experts found the tripartite data sources for maturity score computation beneficial, with

particular approval for the automated pillar due to its facilitation of continuous assessments devoid of manual intervention. They deemed the TSMM content sensible and the grid-style nature very useful, albeit requiring customization to fit the context of each respective organization. Experts stressed that the defined topics and corresponding maturity levels are contingent upon numerous factors, such as employed technology, centrally provided tools and infrastructure, and the resulting responsibilities of individual development teams. Highlighted examples include management of backups for application infrastructure and their data, logging and monitoring practices, utilization and maintenance of third-party tools and libraries, and provision of security testing tools such as static code analysis and vulnerability scanners by central teams.

Experts underscored that the maturity model's topics should not excessively inhibit the autonomy of development teams. For instance, rather than mandating specific techniques for identifying and addressing common vulnerabilities, the assessment could ascertain the presence of such measures, not necessarily their exact form. Discussions also centered on whether team assessment should be primarily based on team capabilities or also incorporate their output, meaning whether an assessment of their developed software products should impact the team maturity score. A majority of experts favored the inclusion of product-specific aspects.

The external assessment pillar was valued for offering a more objective perspective, validating self-assessments, and revealing potential oversights. However, conflicts of interest were noted when team-external roles, like in-house security engineers, were funded by the team's budget. This situation was summarized by an expert as "you do not bite the hand that feeds you", indicating that while security engineers are well-suited for assessments, they may evaluate less critically if financed by the team they are assessing.

Several respondents highlighted the automated pillar's significance, as it offers continuous monitoring without the potentially lengthy gaps of self- or third-party assessments. One expert suggested considering whether teams should be able to override results, such as in cases of false positives, to maintain the relevance of maturity scores and team motivation. However, such overrides should be flagged for transparency. Finally, one interviewee stated that "it is essential to remember that you should not use a sledgehammer to crack a nut", stressing that organizations need to ensure that the scope of the assessment is appropriate and not overly burdensome.

8. Discussion

8.1. Key findings

Our research yielded five key findings which we summarize in the following.

First, the ten proposed team security maturity criteria provide a holistic foundation for creating new maturity models to evaluate the security competency of agile teams. Organizations could also utilize these criteria to assess the completeness of existing models. The expert interviewees concurred with the significance of these factors and did not identify any overlooked critical aspects, though they contributed valuable detailed insights. However, we acknowledge the potential existence of other significant factors not yet identified.

Second, introducing team maturity levels can ameliorate the tension between autonomy and control in LSAD. By fostering transparency and motivating teams to bolster their proficiency in creating secure and compliant applications, central security governance teams can allocate their finite resources more judiciously, e.g., by prioritizing the support of low-maturity teams. Teams with a higher level of maturity, on the other hand, may work more autonomously, thereby better aligning with agile methods. Therefore, the TSMM requires a sustainable integration with security governance, functioning more as a facilitator than a traditional controller. Despite potential initial resistance from central governance and security teams due to concerns of diminished influence and control, we expect that the overall security posture and value creation significantly profit from empowering development teams to improve their security maturity and take greater responsibility. As development teams are closest to their own products, they are best positioned to secure them, given adequate security capabilities. As a result, increasing team maturity also mitigates some of the unique security challenges of LSAD, for example, the alignment of security objectives in distributed settings (van der Heijden et al., 2018). More mature teams require less security coordination and quality assurance.

Third, our evaluations demonstrate a preference among experts for a mixed source approach in calculating team maturity scores, integrating self- and external assessments, and (semi-)automated metrics. While each source may be biased in isolation, their combination yields a holistic perspective.

Fourth, the TSMM offers considerable transparency and feedback, providing insights into an organization's security posture and guiding teams to improve by identifying potential weak spots and areas requiring training. By analyzing team maturity profiles,

organizations can optimize security measures, whether by replacing outdated controls or extrapolating organization-wide actions from teams producing the most secure applications.

Finally, the exact configuration of assessment types, maturity levels, and content of the model is closely linked to the organizational structure, prevailing product risks, and predominant technologies utilized by development teams. Consequently, the main contribution of our TSMM lies less in the specific details of the model and more in its overarching idea and structure. Our primary intent in presenting the TSMM is to inspire organizations to harness the potential of team security maturity and adapt their governance procedures and empower agile teams without compromising security.

The TSMM fills a gap by evaluating maturity of capabilities with a grid-style approach rather than providing a checklist of required practices. It employs descriptive text for classifying maturity levels, aiding self-assessments. This concept could be applied to other concerns of product quality besides security.

8.2. Limitations

In order to adhere to curtail potential research limitations, our study closely followed the empirical standards for software engineering research (ACM SIGSOFT, 2023) during both the interviews and systematic review. The interviews served to evaluate our results. However, given the restrictive time frame of an interview, an exhaustive exploration of each topic along with the practical application of the model within an LSAD environment—particularly observing its impact on factors such as the overall security maturity of development teams—remained beyond the scope of this study. This represents a limitation of our present research, and the exploration of these aspects forms part of our future research agenda.

To improve reliability, all interviews were recorded and transcribed. For the analysis, we conducted systematic, reproducible content analysis and classification as described by Kuckartz (2016). To ensure construct validity (Runeson & Höst, 2009) in our interviews, we employed semi-structured questionnaires, which were first tested for comprehensibility, and ambiguities were clarified directly through dialogues with the interviewees. However, given the conversational nature of the semi-structured approach, potential deviations and imprecisions were inherently difficult to eliminate. Lastly, to scrutinize and offset threats to the validity of our designed artifact ensuing from our DSR process, we utilized the guidelines proffered by Lukyanenko et al. (2014).

9. Conclusion

The rise of scaled agile methods and the growing importance of security create tension between autonomy and control in LSAD. To ease this conflict, we encourage using security maturity scores for development teams. We used DSR to address our RQ on designing a team security maturity model, involving an SLR, expert interviews, and workshops to achieve rigor and practical relevance.

We found ten key criteria from the literature to assess a team's ability to develop secure applications, which we used to create and evaluate the maturity model through expert interviews and workshops. Since our evaluation has been limited to expert interviews, future studies could apply and analyze the TSMM in LSAD settings, examining its effectiveness in measuring team capability and implications on the autonomy-control tension during actual use.

References

ACM SIGSOFT. (2023). *Empirical standards for conducting and evaluating research in software engineering*. Retrieved June 14, 2023, from github.com/acmsigsoft/EmpiricalStandards

Alsaqaf, W., Daneva, M., & Wieringa, R. (2017). Quality requirements in large-scale distributed agile projects – a systematic literature review. *Lecture Notes in Computer Science*, *10153*, 219–234.

AppSecure.nrw. (2021). *Security-belts*. Retrieved June 14, 2023, from github.com/AppSecure-nrw/security-belts

Bartsch, S. (2011). Practitioners' perspectives on security in agile development. *2011 Sixth International Conference on Availability, Reliability and Security*, 479–484.

Becker, J., Knackstedt, R., & Pöppelbuß, J. (2009). Developing maturity models for it management. *Business & Information Systems Engineering*, *1*(3), 213–222.

Bell, L., Brunton-Spall, M., Smith, R., & Bird, J. (2017). *Agile application security: Enabling security in a continuous delivery pipeline*. O'Reilly.

Beznosov, K., & Kruchten, P. (2004). Towards agile security assurance. *Proceedings of the 2004 workshop on New security paradigms*, 47–54.

Bishop, D., & Rowland, P. (2019). Agile and secure software development: An unfinished story.

Bodin, N., & Golberg, H. K. B. (2021). *Software security culture in development teams: An empirical study*. NTNU.

Britto, R., Smite, D., & Damm, L.-O. (2016). Software architects in large-scale distributed projects: An ericsson case study. *IEEE Software*, *33*(6), 48–55.

Dawson, M., Burrell, D., Rahim, E., & Brewster, S. (2010). Integrating software assurance into the software development life cycle (sdlc). *Journal of Information Systems Technology and Planning*, *3*, 49–53.

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.

Döring, N., Bortz, J., Pöschl, S., Werner, C. S., Schermelleh-Engel, K., Gerhard, C., & Gäde, J. C. (2016). *Forschungsmethoden und evaluation in den sozial- und humanwissenschaften*. Springer.

Horlach, B., Böhmann, T., Schirmer, I., & Drews, P. (2018). It governance in scaling agile frameworks. In *Mkwi 2018*.

Jaatun, M. G., & Soares Cruzes, D. (2021). Care and feeding of your security champion. *Int. Conf. on Cyber Situational Awareness, Data Analytics and Assessment*, 1–7.

Julisch, K. (2008). Security compliance. *Proceedings of the 2008 workshop on New security paradigms*, 71.

Kuckartz, U. (2016). *Qualitative inhaltsanalyse. methoden, praxis, computerunterstützung*. Beltz.

Lukyanenko, R., Evermann, J., & Parsons, J. (2014). Instantiation validity in is design research. Springer.

Maier, A. M., Moultrie, J., & Clarkson, P. J. (2012). Assessing organizational capabilities: Reviewing and guiding the development of maturity grids. *IEEE Transactions on Engineering Management*, *59*(1), 138–159.

Nägele, S., Watzelt, J.-P., & Matthes, F. (2022). Investigating the current state of security in large-scale agile development. *23rd Int. Conf. on Agile Software Development (XP)*, 203–219.

Nägele, S., Watzelt, J.-P., & Matthes, F. (2023). *Supplementary material*. Retrieved September 3, 2023, from https://doi.org/10.6084/m9.figshare.c.6819243.v1

OWASP Foundation. (2022). *Owasp samm v2*. Retrieved June 14, 2023, from owaspsamm.org/

Pagel, T. (2020). *Owasp devsecops maturity model*. Retrieved June 14, 2023, from dsomm.timo-pagel.de/

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

Porst, R. (2011). *Fragebogen: Ein Arbeitsbuch*. VS Verlag für Sozialwissenschaften.

Poth, A., Kottke, M., Mahr, T., & Riel, A. (2021). Teamwork quality in technology-driven product teams in large-scale agile organizations. *Journal of Software: Evolution and Process*, e2388.

Poth, A., Kottke, M., & Riel, A. (2020). Scaling agile on large enterprise level with self-service kits to support autonomous teams. *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*, 731–737.

Poth, A., Kottke, M., & Riel, A. (2021). Measuring teamwork quality in large-scale agile organisations evaluation and integration of the atwq approach. *IET Software*, *15*, 443–452.

Rindell, K., Ruohonen, J., Holvitie, J., Hyrynsalmi, S., & Leppänen, V. (2021). Security in agile software development: A practitioner survey. *Information and Software Technology*, *131*, 106488.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164.

Synopsys. (2021). *Building security in maturity model 12 — bsimm*. Retrieved March 1, 2022, from bsimm.com/content/dam/bsimm/reports/bsimm12-foundations.pdf

Uludağ, Ö., Putta, A., Paasivaara, M., & Matthes, F. (2021). Evolution of the agile scaling frameworks. *22nd Int. Conf. on Agile Software Development (XP)*, 123–139.

van der Heijden, A., Broasca, C., & Serebrenik, A. (2018). An empirical perspective on security challenges in large-scale agile software development. *12th ACM/IEEE Int. Symposium on Empirical Software Engineering and Measurement*.

von Solms, R. (1998). Information security management: The code of practice. *Information Management & Computer Security*, *6*(5), 224–225.

Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review.

Wendler, R. (2012). The maturity of maturity model research: A systematic mapping study. *Information and Software Technology*, *54*(12), 1317–1339.