Informatik Spektrum (2023) 46:220–229
https://doi.org/10.1007/s00287-023-01549-5

HAUPTBEITRAG

Practices and challenges of threat modelling in agile environments

Paul Theurich[1] · Josepha Witt[2] · Sebastian Richter[3]

Accepted: 24 August 2023 / Published online: 27 September 2023

Abstract
Facing the increasing annual cybersecurity costs, threat modelling (TM) is a method to consider security as early as possible in the software development life cycle (SDLC). Thereby, TM helps to identify and address security-related design flaws in information systems. As the original TM approach is based on sequential development, it is not aligned with today's predominantly agile environments. This results in several challenges. However, TM's implementation in an agile development approach lacks the recommendations on how to tackle these challenges. Therefore, we assess the state-of-the-art of TM challenges and practices in agile environments by conducting a literature review covering 220 papers. Thereby, we identify nine categories of challenges and six categories of practices. We propose a valuable artefact for practitioners by mapping challenges and practices to the agile SDLC and by creating a matrix highlighting how the practices address the challenges of TM in an agile environment.

Introduction

Many examples in the recent past highlight the increasing importance of information technology (IT) security. The cybersecurity spending worldwide has more than doubled from 2017 to 2022 with annual costs of 71.1 billion US dollars [37], whilst the average cost of a data breach in the US increased in a linear manner from 2014 to 2022 to 9.44 million US dollars [36]. "Insecure Design" being in the fourth place of the latest OWASP Top 10 report [26] underlines the risk originating from security-related design flaws of information systems. As it is beneficial to consider security as early as possible in the software development life cycle (SDLC), threat modelling (TM) is an important starting point to identify architectural flaws [44]. TM aims to identify, describe, and categorise threats [27] and considers potential attackers and their capabilities [8]. Although the importance of TM is well understood in research [8, 10, 15, 27], the original TM approach (cf. based on sequential development) is not aligned with today's predominantly agile environments [12]. Whilst the challenges to use TM in agile environments have been discussed (e.g. [13]), TM's implementation in an agile development approach lacks the recommendations, how to tackle these challenges [10]. Therefore, we assess the state-of-the-art of threat modelling challenges and practices in agile environments. By mapping both to the agile SDLC, we develop a framework which helps practitioners to implement TM in an agile environment.

Methodological approach

To address the research goal, a systematic literature review [47] was conducted and the relevant literature was thoroughly analysed with methods from qualitative research, i.e. coding. We provide the state-of-the-art of TM challenges and TM practices in agile environments from different research perspectives (e.g. information systems, informatics) and outlets (i.e. journals, conference proceedings, and book chapters). Therefore, the literature review covers: a) various search databases (ACM, AISeL, Emerald, IEEE, Springer, Taylor & Francis), b) a broad search with terms combining "Threat Modelling" (cf. BE)/"Threat Modeling" (cf. AE)

Paul Theurich
paul.theurich@mercedes-benz.com

✉ Josepha Witt
josepha.witt@uni-hohenheim.de

Sebastian Richter
sebastian.richter@dhbw-stuttgart.de

[1] Mercedes-Benz Mobility AG, Stuttgart, Germany

[2] Department of Intelligent Information Systems, University of Hohenheim, Stuttgart, Germany

[3] Information Systems, Baden-Wuerttemberg Cooperative State University (DHBW) Stuttgart, Stuttgart, Germany

and "Agile", and c) covers all hits (cf. extensive analysis). The initial result set covered 220 papers. Examining title and abstract and excluding duplicates and papers with other languages than German or English, a set of 57 papers remains. During the full text analysis of the papers, some turned out to be inaccessible or not relevant, resulting in a final set of 35 papers, which were qualitatively analysed in three iterations.

First, the two perspectives of the research question, i.e. challenges and practices, are used as deductive categories (cf. [22]). Relevant text passages are assigned to them. Second, challenges and practices are identified using an inductive coding approach (i.e. open and axial coding, known from Grounded Theory (cf. [38])). Relevant text segments are labelled. Recurring challenges and practices are summarised into categories (cf. approach comparable to open coding). Third, these codes are categorised based on their relationships and combined into overarching codes (cf. approach comparable to axial coding). These three iterations create a hierarchical coding scheme, capable to order the challenges and practices of TM in agile environments. Finally, the categories from the coding scheme are mapped to the agile SDLC.

Threat modelling

The purpose of TM is the identification of potential threats incorporated by the design of the application. Such threats harm one or more tangible (e.g. customer data) or intangible (e.g. corporation image) assets. Hence, the goal is to protect an asset from the respective threat [7]. Threats can be identified in three steps: 1) identification of assets to be protected; 2) analysis how assets are stored, handled, and processed, e.g. by creating a data flow diagram (DFD); 3) definition and analysis of threats that affect assets [10]. TM includes to define the threat response which can be classified. Threats can be accepted (cf. small likelihood of its exploitation and/or small impact), mitigated (i.e. taking measures to reduce/eliminate the chance of exploitation), or avoided (e.g. by altering an application's architecture) [7].

The creation of a DFD is one of the initial steps of TM [15, 34]. A DFD visualises the information flows between users, components, and systems, whilst taking into account changes of data as well as its storage [15]. Furthermore, the transfer of information (e.g. protocol) is part of the DFD. Trust boundaries within the DFD, such as the way from intranet of a company to the public internet, make changes of the trust level transparent [7].

One advantage of asset-based TM is that it can start before the application or system is developed. The method guides developers and creates transparency of crucial and protect-worthy assets. This shifts security as an afterthought, addressed by security measures during development, to a security-by-design approach [30]. However, its origin in sequential development processes implies several challenges in todays' prevalent agile environments.

Challenges of threat modelling in agile environments

As follows, the literature-based challenges of threat modelling in an agile environment are described along the identified categories (C1–C9) and mapped to the agile SDLC (cf. Fig. 1). Agile SDLC is a generalised process based on the common ground of various agile development methods (e.g. Scrum, Extreme Programming, Dynamic System Development) described in comparative studies (e.g. [2, 11]). The first phase of agile SDLC is the *vision, exploration, and inception phase*, which covers a systems' general objectives as well as user-based requirements, the definition of the project team and its responsibilities as well as other resources and tools. The software is developed in several iterations of *construction* and *release* comprising traditional SDLC phases (i.e. *plan, design, and develop, test, and deploy*). The life cycle ends with the *maintenance phase* of the final software product. In the following, the challenges are explained, ordered by the aforementioned phases.

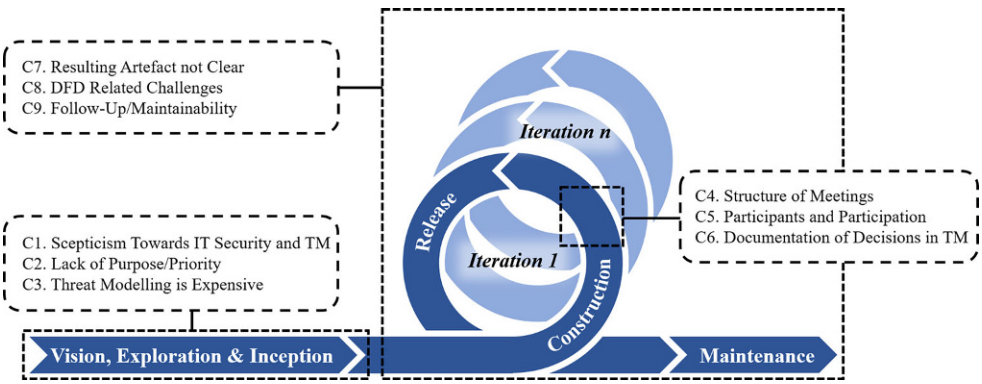

Fig. 1 Categories of threat modelling (*TM*) challenges along the agile software development life cycle

Challenges affecting the vision, exploration, and inception phase

C1. Scepticism towards IT security and TM Challenge C1 deals with scepticism towards IT security in general or towards TM, especially in an agile environment. The latter lies inter alia in the fact that *TM has its roots in waterfall-like development*. Traditional TM frameworks like Microsoft's Security Development Lifecycle (MS SDL) are not fully compatible to an agile environment [12]. Whereas traditional software security practices require extensive planning and analysis, this is not feasible in an agile environment; requirements typically change often [21]. The detailed picture of the target system is not anymore sought-after at the beginning of the project. Instead it emerges due to the iterative agile approach [6]. In the past, TM was conducted once, before the implementation had begun. This contradicts the agile approach of continuous evolution and integration in countless iterations [33].

At first glance, one might argue, that TM contradicts the agile principles as such. Principles as "Individuals and interactions over processes and tools", "Working software over comprehensive documentation", or "Responding to change over following a plan" might support that argument [9]. However, that there is no need to document is a misunderstanding. This results in developer's loss of focus to tackle security-related documentation (i.e. decisions, risks, and assets) [42]. According to Baca and Carlsson [6], Microsoft's TM approach contradicts the agile manifesto as collaboration and small teams shall be prioritised over writing secure tools. Standard TM frameworks, like MS SDL (https://www.microsoft.com/en-us/securityengineering/sdl), originating from waterfall development, scale badly in small and flexible agile environments.

In case no issues and threats were identified during the threat modelling session the impression might come up, that there is no need to worry about security [13]. This may lead to a *false sense of security*. Furthermore, developers show *scepticism towards prescriptive methodologies and formal routines* found in traditional quality systems and frameworks [49].

C2. Lack of purpose/priority Agile teams often run a non-agile security development life cycle parallel to the agile development process. This roots in the *conflict of interest*, since security requirements are not a responsibility of the development teams [28]. Without the alignment of security and business objectives, product owners prioritise business value over security. This leads to a conflict between security experts, product owners, developers, and the management. One reason for this is, that the business case for security is not transparent [43].

Furthermore, there can be a *lack of motivation* to conduct security practices, e.g. creating a DFD. Participants of DFD-sessions could feel overwhelmed or might want to use the time for development activities instead [13]. TM sessions are likely being skipped altogether if the developers are not motivated and do not see their purpose [10].

Furthermore, the *responsibilities* for security are often *unclear*, which results in a negligence and a lack of priority [43]. Identified threats in product components, which already passed the definition of done, are seldom prioritised due to time concerns [13].

C3. Threat modelling is expensive As TM activities are time-intensive, manual tasks, practitioners link TM to *high effort* [43] and perceive it as a costly activity [6]. The same applies to the definition of the software architecture [21]. Due to the frequency of changes and software delivery in an agile SDLC, developers do not have time to invest great effort into risk analysis and TM during the development phase [12].

In practice, TM is often deemed *not worth the effort* [10]. Frameworks like MS SDL, McGraw Touchpoints and Common Criteria practices appear to be too expensive whilst bringing insufficient value to be integrated into the agile development process. Looking at their individual practices, it might make sense to adopt them [24]. However, due to the high effort in combination with cost pressure, TM is often left out [19].

Challenges affecting single meetings in an iteration

C4. Structure of meetings TM also lacks a *structure to conduct a TM meeting* efficiently within a sprint as a clear recommendation on how to start the process (e.g. looking at DFD, following STRIDE[1], ad-hoc) [13, 19]. During a TM session, the group can get lost in details and focus. These *unfocussed and unproductive meetings and discussions* are inefficient. Allocating 1 or 2 hours in a sprint to conduct TM is a major time investment. Not covering all the planned topics, e.g. analyse the complete DFD in one session, might frustrate developers [13].

C5. Participants and participation First, it is a challenge to *identify the essential participants* of a TM session, especially in projects consisting of multiple product teams.

[1] Microsoft's STRIDE [32] offers a chart, which summarises a set of threats (STRIDE stands for spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege) and is a method to brain-storm threats [44].

Furthermore, whether the involvement of the product owner is beneficial for the TM process is rather unclear [13].

During the actual TM session, *not all attendees might actively participate*. On the one hand, attendees might lack the overall picture, e.g. knowledge of every part of the DFD, especially when *looking at cross-product systems*. On the other hand, it can be hard to motivate the participants [13]. A collaboration-related challenge can be the *TM communication channel*. The combination of physical and remote attendance (cf. video conference) can result in inefficient sessions [13]. This is especially relevant if development teams are geographically separated [17].

Another challenge regarding the participants is the *lack of expertise* in security, security governance and software development in an agile environment when conducting the individual steps of TM, e.g. the risk analysis [19, 25]. Developers might not have received sufficient software security instructions or training [17]. Even a security champion of a team (i.e. a team member who is enthusiastic about and advocates for security [39]) might not have enough experience to guide through a TM session [10]. Therefore, often security experts are required which lead the meeting and provide guidance [43] as well as boost creativity to define relevant attack scenarios [13].

C6. Documentation of decisions in threat modelling A challenge is that working sessions and their results (e.g. the assets identified) are *not properly documented* [13] as this hinders future TM sessions. First, not having a pre-defined design or architecture, which serves as a foundation, has a reinforcing effect as it increases the effort to start TM [17]. Second, the process of a TM session, i.e. *the assumptions made, should be documented* as excluded threats based on assumptions might turn out to be relevant later [46]. Furthermore, architectural assumptions should be transparently documented and managed since they are an important form of implicit (if not documented) architectural knowledge. Hence, not documented assumptions can hinder their systematic reuse, the evolution of the system, as well as the integration of TM in an agile environment. Looking at systems' attacks it appears that assumptions during the development phase can result in design flaws [46]. Also, DFD-related limitations regarding undocumented decisions affecting architecture and security can lead to assumptions. This might result in redundant information in the threat model and therefore more effort, or undocumented assumptions, which might result in overlooked threats [33].

Challenges affecting the process of TM in various iterations

C7. Resulting artefact not clear The *output* of a TM session might *not* be *concrete enough*. For example, impact and probability of threats or their mitigation are not concretely modelled [13]. How to document the result and outcome of a TM session is often unknown by practitioners [17]. This challenge can stretch over multiple iterations of the agile SDLC and it may be *unclear when to finish the analysis*. Discussing every aspect of the DFD in each iteration can take too long. Having frequent iterations, only changing aspects can be discussed, with the risk to miss a dependency, which changed as well [13]. The definition of done of TM is often not defined [10].

Also, it can be challenging to find an appropriate *level of granularity* in TM, e.g. when creating a DFD. If the DFD is to abstract, it is an insufficient basis for the purpose of TM. If the granularity is too fine, the creation takes too long and meetings become inefficient [13].

C8. Data flow diagram-related challenges A challenge is the separation of the DFD from the actual code. For example, looking at legacy systems, an overview in form of a DFD does not provide confidence, since it does not necessarily reflect the actual way the system works. A *link to the actual code is missing* [13].

Projects consisting of many different products, accounted by several teams, are hard to visualise in a (fine-grained) DFD. Also, inviting all parties involved to a TM and DFD creation session can be hard to achieve. The *definition of boundaries to systems of other teams* is an additional challenge [13].

C9. Follow-up/maintainability *Keeping* the created *documentation up to date* is also a challenge. Depending on the scale of the system, it is challenging to analyse and maintain the threat model through the iterations, as TM should be performed every iteration (cf. regularly reviewed) during the requirement and design phase [28]. Also, if the assets or DFD are not properly documented, they are soon deprecated. There is a lack of motivation to frequently analyse the existing DFD and corresponding system changes. If they are not updated frequently, it is hard to recall all changes which might lead to a wrong representation of the system. If the updates are too frequently, developers might lose interest [13].

Furthermore, the *follow-up of the threats* is a challenge. It might be necessary to describe threats in more detail to use the documentation valuably. Also, *threats might change, or appear over time*, which can make it hard to keep them current [13].

Practices of threat modelling in agile environments

Analysing the literature, we identified six categories of TM practices in an agile environment (P1–P6).

P1. Promote and teach security As the integration and tackling of security in the agile SDLC takes time and will slow down the output rate of new functionalities, new metrics for the developer's performance are required to incentivise them to address security. Hence, the business goals need to be aligned with security concerns. If security concerns are valued, security is more likely to be integrated in the agile SDLC [13].

Apart from transparency and alignment, developers, product owners, and other team members should be sufficiently trained [28]. Besides traditional training sessions, gamification of TM, e.g. by using Microsoft's Elevation of Privilege (EoP) can be used. Although EoP is discontinued from Microsoft for the actual TM of software artefacts, it is still used for education [41]. EoP shows positive effects on sensitising software security and starting discussions; developers want to improve systems' security as well as security know-how [41].

P2. Structuring of meetings TM meetings can be structured by the way they are conducted or by determining the participants. As follows, we briefly describe some approaches for conducting such meetings and recommend to gather further information from the original papers before implementation.

Weir et al. [49] suggest a lightweight threat assessment workshop to analyse possible threat actors and outcomes. The workshop mainly addresses shortcomings in know-how regarding potential technical threats and the availability of professionals in this field by providing room for the participants to brainstorm in an ideation session. Tøndel et al. [43] propose a regular security intention recap meeting (with TM as one activity) to discuss the current state of security of an application (i.e. security goals and continuous self-evaluation indicators), and to make security-related decisions transparent. Thereby, it should be ensured that each iteration results in actionable decisions which are being addressed during development. Tøndel et al. [42] introduce the Protection Poker game, which shall be played in the planning phase of each iteration of the agile SDLC. The goal is to rank security risks for features and to define suitable security measures, to ensure that the risk remains acceptable. Whilst Protection Poker supports the incremental and continuous approach of security and engages the whole team with security-related issues concretely, it is time-consuming and not widely used. De [14] proposes a TM approach based on design thinking and Scrum, which aims to provide a practical way of tackling TM. It suggests a deep-dive into the system and its sub-components to identify threats and ways to mitigate them.

Besides these approaches on how to conduct TM, several papers highlight the importance of certain participants. Developers are required when identifying potential threats [10, 19]. Security experts structure meetings by scheduling and preparing them, creating a DFD, documenting the identified threats and risks throughout the meeting [13, 17, 19].

P3. Structuring the process Practices that structure the process do not refer to the structure of a single TM session, but to the integration of TM in an agile SDLC, consisting of numerous TM sessions in many iterations.

During the early stages of the development phase, a first threat model can be created, which increases in granularity and quality over time and iterations [29]. Later, TM focuses on changes between two iterations. This can result in a prioritisation of a specific part of the system in the next code review session or a targeted penetration test [25]. This continuous approach accepts that new threats might appear over time, whilst others might be mitigated [48]. To avoid overhead and rapid increase of effort, Kus and Sohr [19] suggest when to conduct TM in an iteration (cf. extension of STRIDE). TM iterations are triggered after a change of: a) the business process, b) the trust-boundaries, c) the interfaces and attack surface, or d) the protect-worthy assets.

Apart from the iterative structure, attacker-centric/abuser stories can be created in agile working sessions to identify related attacker entry points and address them in the following sprints. This approach requires attacker personas, developed during a profiling exercise or shared within the organisation and across teams [4]. The usefulness of persona creation might be limited due to time constraints and has to be evaluated in practice [23]. The product owner creates these stories and during the sprint, the developers tackle them.

Another approach is to treat TM as a story itself. The required tasks can be split to allow an estimation of effort and to add them into the backlog of the product. Templates can save time and standardise the outcome [21]. Furthermore, the outcomes of a TM session can be included in security-related stories, which allows a prioritisation based on the risk [19].

P4. Threat modelling models The literature captures different models how to conduct TM in an agile environment. In comparison to P2 and P3, this category introduces predefined, holistic schemes of activities to follow when conducting TM.

As DFD is an important part of TM, several papers suggest to enhance the DFD notation. On the one hand, secu-

rity information can be documented in the DFD by adding security properties (e.g. specifying the transferred information regarding encryption) and security mechanisms (e.g. key management solutions and authentication mechanisms) [45]. This approach can reduce the amount of undocumented assumptions [46]. On the other hand, the DFD Meta Model can be enhanced regarding its notation (e.g. adding elements such as data storage and trust boundaries), and the extension to threat types/catalogues and integrated security solutions [33].

The Visual, Agile, and Simple Threat (VAST) model offers a scalable and automatable TM solution suitable in an agile development setting. VAST supports the whole SDLC [1] and consists of two models. The operational threat model aims to provide insights to the infrastructure by offering a DFD from attacker perspective. The application threat model maps functionality and communication between processes in a process flow diagram to support developers [18].

Process for Attack Simulation and Threat Analysis (PASTA) is a risk-centric TM approach, including the analysis of abuse cases and design flaws, as well as modelling and simulating attacks [18].

OWASP Application Threat Modelling supports different target groups by offering three adapted approaches: an attack-centric approach (for security testers), a system-centric method (for developers), and an asset-centric view (for security managers). However, this impedes to get the overall picture and consider all risks [29].

P5. Security engineering process The security engineering process (SEP) supports the delivery of secure software by offering activities related to the development, maintenance, and delivery of the product. Therefore, this category of practices refers to high-level approaches that address security in the SDLC, adapted to agile environments [5]. Hence, the introduced practices cover TM as one process step of many.

Microsoft's Security Development Lifecycle (MS SDL) offers a lightweight, tailored approach for agile environments (MS SDL/A) which addresses TM during the design phase [20].

The Comprehensive, Lightweight Application Security Process (CLASP) offered by OWASP includes supportive activities for TM, e.g. detail misuse cases, perform security analysis of system requirements and design, identify attack surface, and apply security principles to design [31].

The meta-model MetaSEnD consists of practices established in Secure Software Development Lifecycles (SSDLCs), standards, testing tools, and certifications. As in most other models, TM and the following risk analysis are continuous tasks of the design phase [16].

The Secure Scrum Process (cf. [21]) describes seven activities related to risk analysis, conducted in requirement identification and design phase, e.g. the identification of external dependencies and trust levels (cf. users access rights). Furthermore, this approach introduces security-related user stories with certain debt value based on the security benefit of the story, or the risk it mitigates. The development team is incentivised to regularly tackle the security stories as if the overall security debt exceeds a pre-defined threshold, a whole sprint shall be dedicated to the security stories in the backlog [21].

Apart from introducing another pre-defined security engineering approach, Sharma and Bawa [31] suggest building an own framework for secure agile development based on existing SEP frameworks (e.g. MS SDL, CLASP) adapted to specific project needs.

P6. Tool support Tool automation supports the time intensive TM processes.

Dataflow-based tool approaches follow frameworks such as STRIDE, VAST, or PASTA, and analyse the DFD to identify patterns which are linked to a set of threats. Examples include MS TM, OWASP Threat Dragon, and Waypoint. This approach is suitable for analysing pure software applications, but provides only limited support for cyber-physical systems, merging hardware and software [40]. Kus and Sohr [19] propose an automated way of creating DFDs by conducting static analysis and dynamic analysis (cf. runtime information) of the source code and the configuration of the system. Based on this, Tuma et al. [45] explored an automated analysis of the DFD to detect security design flaws. However, the low result precision compared to manual TM approaches highlight the necessity of manual TM. The notation of the DFD regarding security-related measures and mechanisms need to be further enhanced to enable automated tool support following a dataflow-based approach.

In contrast, attack graph-based approaches simulate the exploitation of vulnerabilities by forming a chain of subsequent attacks to reach a specific goal. This attack graph, traditionally manually build by security experts, can be created automatically by tools, such as CORAS, MulVal, and TVA. Unfortunately, these tools are not able to cover the complete variety of all potential attacks, such as zero-day vulnerabilities [40]. Other tools such as ADTool, Mobius, and CyberSAGE combine dataflow-based and attack graph-based approaches (cf. hybrid model) and are therefore suitable for complex systems and critical infrastructure [40]. Althar et al. [3] propose to predict threats utilising machine learning approaches based on a growing knowledge system. This system supports the TM process by offering an automated prediction model for threat assessment based on the requirements of the customer, which are translated into Common Weakness Enumeration (CWE) categories.

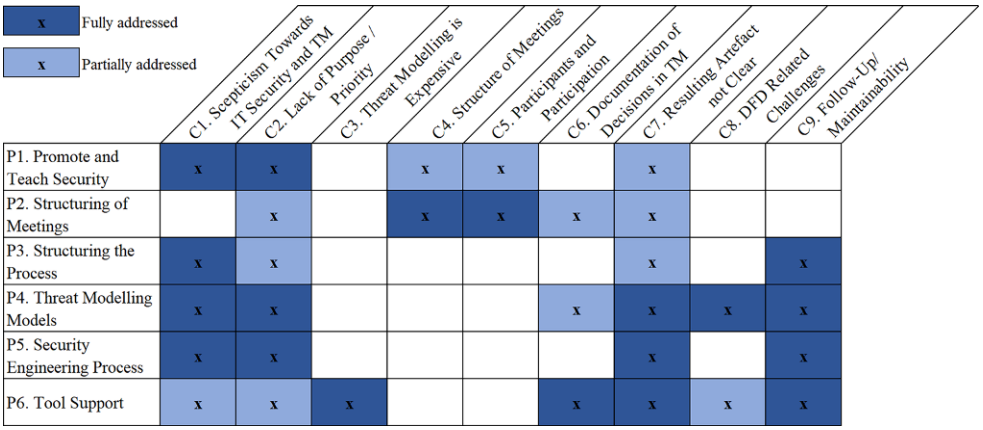

Fig. 2 Mapping of threat modelling (*TM*) challenges and practices

Continuous Threat Analysis and Management (*CTAM*) aims to make threat analysis, threat management and progress monitoring a continuous task by automating it in the continuous integration pipeline. CTAM achieves this by: 1) connecting the source code to the DFD model, 2) identifying threats and their risk analysis for every push to the repository by conducting a continuous integration analysis, 3) gathering and summarising the results and findings in the CTAM server, and 4) offering this information to the developers. It motivates the developers to keep the architectural abstraction model of the application up to date [35].

Discussion

This section maps the identified practices and challenges of TM in an agile environment by highlighting how the practices mitigate the challenges. This concludes with a mapping matrix, illustrated by Fig. 2. Furthermore, the implications of the mapping are discussed.

To promote security (cf. P1), the benefits of the integration of security in the agile SDLC should be made transparent to developers, product owners, and management such that the business goals are aligned with security concerns (cf. C2). In addition, such practices address the scepticism and false sense of security (cf. C1). Furthermore, the knowledge transferred in training sessions helps to conduct productive meetings (e.g. increasing focus and productivity; cf. C4) and to determine the participants as well as their tasks (cf. C5). Furthermore, training increases experience. Experienced and trained personnel is capable to develop concrete results with an appropriate level of abstraction and can decide when to stop the analysis (cf. C7).

The approaches introduced provide a certain structure for TM meetings (cf. P2, C4) and clarify the need for and role of certain participants (cf. C5). Especially the attendance of developers and security experts is highlighted. Consequently, the expertise of developers and security experts can support the development of concrete result on a certain level of abstraction (cf. C7). Structured meetings (e.g. the regular security intention recap meeting) help to bring transparency into security-related decisions (cf. C6) and ensure that each iteration results in actionable decisions (cf. C7). Furthermore, many approaches (e.g. Protection Poker) address the lack of motivation and security awareness by engaging the whole team and assume responsibility for the TM in general (cf. C2).

The integration of TM in iterations (cf. P3) ensures compatibility with the agile manifesto and therewith alleviates the scepticism towards TM in an agile environment (cf. C1). Furthermore, the follow-up of threats and the consideration, that threats might change or reappear over time, is fully addressed by conducting TM in iterations and based on defined triggers (cf. C9). By creating stories, to be tackled by developers in the next iteration, the lack of purpose/priority is partially addressed, especially with regard to the transparency of responsibilities within the process (cf. C2). Furthermore, attack-centric stories enable guidance through a structured process and, thereby, lead towards a more concrete result of TM (cf. C7).

With their holistic approach, TM models (cf. P4) address challenges occurring along the whole agile SDLC. Their pre-defined schemes ensure that TM is applicable in agile environments (cf. C1), address the lack of motivation, and list responsible persons (cf. C2). In addition, their prescribed process structure addresses the documentation of performed activities (cf. C6) and leads teams towards a clear result (e.g. documents, threat model) from TM sessions (C7). The iterative structure of these holistic TM models ensures that the results of TM sessions are maintained regularly (cf. C9). Furthermore, several models suggest enhancements of the DFD notation to address the DFD notation challenges (cf. C8).

Several SEP models exist (cf. P5) that embrace a full picture of the software development process and show that TM as well as IT security concerns in general can be applied in an iterative and continuous approach (cf. C1). SEP

offers models and tools which support developers to engineer a secure product in an agile environment, and thereby address the lack of responsibility; developers have a focus regarding purpose and priority (cf. C2). Additionally, expected results of individual process steps are transparent and it is clarified by predefined schemes and tools when the analysis stops (cf. C7). The iterative and continuous approach of SEP addresses the follow-up, the alteration or reappearance of threats over time (cf. C9).

An automated tool support (cf. P6) reduces the effort of TM and improves the cost-to-benefit ratio. Hence, it directly addresses the challenge that TM is expensive (cf. C3). Furthermore, automated approaches achieve a repeatable and concrete result (cf. C7) and security-related assumptions of the product are logged (cf. C6). Threats and documentations are continuously maintained through the iterations (e.g. iterative and automated DFD creation) (cf. C9). Further, the automation of DFD creation mitigates the conflict of interest of developers and their lack of motivation to conduct time intensive security tasks (cf. C2). Finally, the continuity of e.g. CTAM eases TM to be conducted in an agile setting (cf. C1).

The following mapping matrix in Fig. 2 depicts the mapping of the challenges identified and the respective practices by examining which practices fully address the challenges (blue plus 'x'), partially address them (light blue plus 'x'), or do not address them (blank).

Looking at the mapping matrix, it seems that each of the challenges identified is addressed by respective practices. However, each of the practices have dependencies with others (e.g. tool support requires training) which impedes its implementation. Therefore, simply implementing one or more of the mentioned practices will not necessarily lead to a successful TM approach. Rather the specific challenges of a team have to be identified and a custom approach to conduct TM, consisting of certain practices, has to be defined and put into practice.

Furthermore, due to the iterative nature of the agile SDLC, the implementation of a TM approach cannot be done at once. Rather a method, that is not yet described in the literature, is required, which explains how to embed TM as a holistic process into agile project management approaches. By doing so, interdependencies of TM practices and agile project management activities will emerge and raise new impediments and challenges.

Conclusion

Using a systematic literature review and classification techniques, we identified nine categories of challenges and six categories of practices considered to TM in an agile environment. The challenges are matched with the phases of the agile SDLC (cf. Fig. 1). The practices are discussed based on the challenges they address (cf. Fig. 2). Many practices address a general scepticism against IT security and TM (cf. C1) and a lack of purpose and priority (cf. C2). But only the practice to promote ad teach security (cf. P1) addresses directly the vision, exploration, and inception phase. Different approaches to structure TM sessions (cf. P2) face challenges in single meetings (cf. C4–C6). More holistic approaches structure the process of TM in several iterations (cf. P3–P5) on different levels, i.e. focused on TM or the whole security engineering process. Across these levels, TM is conducted incrementally, whilst the risk analysis is done in the design phase. Tools can support the automation (cf. P6) of certain steps along the whole TM process, but also address the high effort and costs for TM directly (cf. C3). The mapping of challenges and practices to the agile SDLC is illustrated in Fig. 3.

The proposed artefacts (cf. matrix and mapping to the agile SDLC) are valuable for practitioners and researchers alike. Practitioners are therewith able to map their own challenges with the presented categories to assess practices

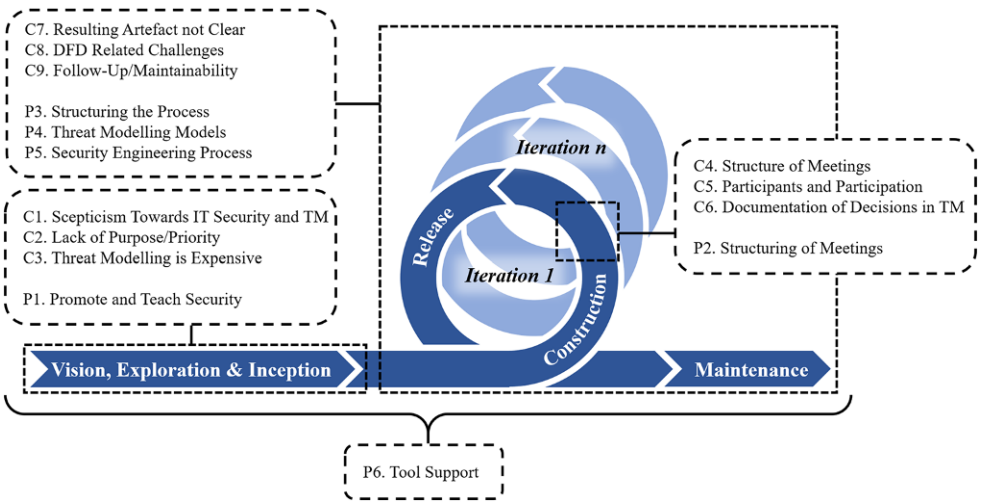

Fig. 3 Categories of threat modelling (*TM*) challenges and practices along the agile software development life cycle

which might be helpful in their case. They are able to justify certain practices as they understand which challenges can be addressed or prevented. Researchers can benefit from a comprehensive review of the state-of-the-art in this field to gain knowledge or conduct further research based on the findings. For example, the presented matrix can be helpful to develop and propose a comprehensive consulting method that utilises existing practices and therewith addresses most challenges. Furthermore, further research can address shortcomings of this paper such as to evaluate the completeness and usefulness of the matrix as an artefact. For example, this can be assessed by conducting a case study, e.g. by applying the matrix in an organisation that seeks help in applying threat modelling in an agile development approach.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Alevizos L, Stavrou E (2022) Cyber threat modeling for protecting the crown jewels in the Financial Services Sector (FSS). Inf Secur J Glob Perspect. https://doi.org/10.1080/19393555.2022.2104766
2. Alsaqqa S, Sawalha S, Abdel-Nabi H (2020) Agile software development: methodologies and trends. Int J Interact Mob Technol 14(11):246. https://doi.org/10.3991/ijim.v14i11.13269
3. Althar RR, Samanta D, Kaur M, Singh D, Lee H-N (2022) Automated risk management based software security vulnerabilities management. IEEE Access 10:90597–90608. https://doi.org/10.1109/ACCESS.2022.3185069
4. Anwar Mohammad MN, Nazir M, Mustafa K (2019) A systematic review and analytical evaluation of security requirements engineering approaches. Arab J Sci Eng 44(11):8963–8987. https://doi.org/10.1007/s13369-019-04067-3
5. Ayalew T, Kidane T, Carlsson B (2013) Identification and evaluation of security activities in agile projects. In: Proceedings of the 18th nordic conference on secure IT systems (Nordsec) Ilulissat, 18. Oct.–21. Oct, pp 139–153 https://doi.org/10.1007/978-3-642-41488-6_10
6. Baca D, Carlsson B (2011) Agile development with security engineering activities. In: Proceeding of the 2nd workshop on Software engineering for sensor network applications (SESENA) Waikiki, 21. May–22. May, pp 149–158 https://doi.org/10.1145/1987875.1987900
7. Baker M (2022) Threat modelling. In: Baker M (ed) Secure web application development. Apress, Berkeley, pp 43–58
8. Bass L, Holz R, Rimba P, Tran AB, Zhu L (2015) Securing a deployment pipeline. In: Proceedings of the 3rd International Workshop on Release Engineering (RELENG) Florence, 19.05., pp 4–7 https://doi.org/10.1109/RELENG.2015.11
9. Beck K, Beedle M, van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, Kern J, Marick B, Martin RC, Mellor S, Schwaber K, Sutherland J, Thomas D (2001) Manifesto for agile software development. https://agilemanifesto.org/. Accessed 9 Oct 2022
10. Bernsmed K, Jaatun MG (2019) Threat modelling and agile software development: Identified practice in four Norwegian organisations. In: Proceedings of the International Conference on Cyber Security and Protection of Digital Services (Cyber Security) Oxford, 03. Jun.–04. Jun, pp 1–8 https://doi.org/10.1109/CyberSecPODS.2019.8885144
11. Bhalerao S, Puntambekar D, Ingle M (2009) Generalizing agile software development life cycle. Int J Comput Sci Eng 1(3):222–226
12. Clark S, Collis M, Blaze M, Smith JM (2014) Moving targets. Security and rapid-release in Firefox. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS) Scottsdale, 03. Nov.–07. Nov, pp 1256–1266 https://doi.org/10.1145/2660267.2660320
13. Cruzes DS, Jaatun MG, Bernsmed K, Tøndel IA (2018) Challenges and experiences with applying Microsoft threat modeling in agile development projects. In: Proceedings of the 25th Australasian Software Engineering Conference (ASWEC 2018) Adelaide, 26. Nov.–30. Nov., pp 111–120 https://doi.org/10.1109/ASWEC.2018.00023
14. De S (2020) A novel perspective to threat modelling using design thinking and agile principles. In: Proceedings of the 6th International Conference on Parallel, Distributed and Grid Computing (PDGC) Waknaghat, 06. Nov.–08. Nov., pp 31–35 https://doi.org/10.1109/PDGC50313.2020.9315844
15. Faily S, Iacob C (2017) Design as code: facilitating collaboration between usability and security engineers using CAIRIS. In: Proceedings of the 25th International Requirements Engineering Conference Workshops (REW) Lisbon, 04. Sept.–08. Sept, pp 76–82 https://doi.org/10.1109/REW.2017.23
16. Granata D, Rak M, Salzillo G (2022) MetaSenD: a security enabled development life cycle Meta-model. In: Proceedings of the 17th International Conference on Availability, Reliability and Security (ARES) Vienna, 23. Aug.–26. Aug, pp 1–10 https://doi.org/10.1145/3538969.3544463
17. Jaatun MG (2019) Architectural Risk Analysis in Agile Development of Cloud Software. In: Proceedings of the International Conference on Cloud Computing Technology and Science (CloudCom) Sydney, 11. Dec.–13. Dec, pp 295–300 https://doi.org/10.1109/CloudCom.2019.00050
18. Kaur G, Habibi Lashkari Z, Habibi Lashkari A (2021) Cybersecurity threats in Fintech. In: Kaur G, Habibi Lashkari Z, Habibi Lashkari A (eds) Understanding cybersecurity management in Fintech (future of business and finance). Springer, Cham, pp 65–87
19. Kus M, Sohr K (2020) Praktische Erfahrungen und Ansätze für âSecurity by Design‘ auf Basis der STRIDE-Methodik. Datenschutz Datensich 44(11):750–754. https://doi.org/10.1007/s11623-020-1361-6
20. Lipner S (2010) Security development lifecycle. Datenschutz Datensich 34(3):135–137. https://doi.org/10.1007/s11623-010-0021-7
21. Maier P, Ma Z, Bloem R (2017) Towards a Secure SCRUM Process for Agile Web Application Development. In: Proceedings of the 12th International Conference on Availability, Reliability and Security (ARES) Reggio Calabria, 29. Aug.–01. Sept, pp 1–8 https://doi.org/10.1145/3098954.3103171

22. Mayring P (2014) Qualitative content analysis: theoretical foundation, basic procedures and software solution. Social Science Open Access Repository (SSOAR), Klagenfurt
23. Moeckel C (2020) Attacker-centric thinking in security. perspectives from financial services practitioners. In: Proceedings of the 15th International Conference on Availability, Reliability and Security (ARES) Virtual Event, 25. Aug.–28. Aug, pp 1–10 https://doi.org/10.1145/3407023.3407082
24. Morrison P, Smith BH, Williams L (2017) Surveying Security Practice Adherence in Software Development. In: Proceedings of the Hot Topics in Science of Security: Symposium and Bootcamp (HoTSoS) Hanover, 04. Apr.–05. Apr, pp 85–94 https://doi.org/10.1145/3055305.3055312
25. Nägele S, Watzelt J-P, Matthes F (2022) Investigating the Current State of Security in Large-Scale Agile Development. In: Proceedings of the 23rd Agile Processes in Software Engineering and Extreme Programming (XP) 13. Jun.–17. Jun, pp 203–219 https://doi.org/10.1007/978-3-031-08169-9_13
26. (2021) OWASP: A04:2021—Insecure Design. https://owasp.org/Top10/A04_2021-Insecure_Design/. Accessed 15 July 2023
27. Rahman AAU, Williams L (2016) Software Security in DevOps: Synthesizing Practitioners' Perceptions and Practices. In: Proceedings of the International Workshop on Continuous Software Evolution and Delivery (CSED) Austin, 14. May–15. May
28. Rindell K, Hyrynsalmi S, Leppänen V (2018) Aligning security objectives with agile software development. In: Proceedings of the 19th International Conference on Agile Software Development: Companion (XP) Porto, 21. May–25. May, pp 1–9 https://doi.org/10.1145/3234152.3234187
29. Rohr M (2018) Sicherheitsuntersuchungen von Webanwendungen. In: Rohr M (ed) Sicherheit von Webanwendungen in der Praxis. Springer, Wiesbaden, pp 345–431
30. Shaked A, Reich Y (2021) Model-based Threat and Risk Assessment for Systems Design. In: Proceedings of the 7th International Conference on Information Systems Security and Privacy (ICISSP) 11. Feb.–13. Feb, pp 331–338 https://doi.org/10.5220/0010187203310338
31. Sharma A, Bawa RK (2022) Identification and integration of security activities for secure agile development. Int J Inf Tecnol 14(2):1117–1130. https://doi.org/10.1007/s41870-020-00446-4
32. Shostack A (2007) STRIDE chart. Microsoft Security. https://www.microsoft.com/en-us/security/blog/2007/09/11/stride-chart/. Accessed 31 Oct 2022
33. Sion L, Yskout K, van Landuyt D, Joosen W (2018) Solution-aware data flow diagrams for security threat modeling. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing (SAC) Pau, 09. Apr.–13. Apr, pp 1425–1432 https://doi.org/10.1145/3167132.3167285
34. Sion L, Tuma K, Scandariato R, Yskout K, Joosen W (2019) Towards automated security design flaw detection. In: Proceedings of the 34th International Conference on Automated Software Engineering Workshops (ASEW) San Diego, 11. Nov.–15. Nov, pp 49–56 https://doi.org/10.1109/ASEW.2019.00028
35. Sion L, van Landuyt D, Yskout K, Verreydt S, Joosen W (2021) Automated threat analysis and management in a continuous integration pipeline. In: Proceedings of the Secure Development Conference (SecDev) 18. Oct.–20. Oct, pp 30–37 https://doi.org/10.1109/SecDev51306.2021.00021
36. (2022) Statista: Average cost of a data breach in the United States from 2006 to 2022. (in million U.S. dollars). https://www.statista.com/statistics/273575/us-average-cost-incurred-by-a-data-breach/. Accessed 15 July 2023
37. Statista (2023) Spending on cybersecurity worldwide from 2017 to 2022. (in billion U.S. dollars). https://www.statista.com/statistics/991304/worldwide-cybersecurity-spending/. Accessed 15 July 2023
38. Strauss AL (1987) Qualitative analysis for social scientists. Cambridge University Press
39. Tahaei M, Frik A, Vaniea K (2021) Privacy champions in software teams: understanding their motivations, strategies, and challenges. In: Proceedings of the Conference on Human Factors in Computing Systems (CHI) Yokohama, 08. May–13. May, pp 1–15 https://doi.org/10.1145/3411764.3445768
40. Temple WG, Wu Y, Cheh C, Li Y, Chen B, Kalbarczyk ZT, Sanders WH, Nicol D (2023) CyberSAGE: the cyber security argument graph evaluation tool. Empir Software Eng 28(1):1–35. https://doi.org/10.1007/s10664-021-10056-8
41. Tøndel IA, Oyetoyan TD, Jaatun MG, Cruzes D (2018) Understanding challenges to adoption of the Microsoft elevation of privilege game. In: Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security (HoTSoS) Raleigh, 10. Apr.–11. Apr., pp 1–10 https://doi.org/10.1145/3190619.3190633
42. Tøndel IA, Jaatun MG, Cruzes DS, Williams L (2019) Collaborative security risk estimation in agile software development. ICS 27(4):508–535. https://doi.org/10.1108/ICS-12-2018-0138
43. Tøndel IA, Cruzes DS, Jaatun MG, Rindell K (2019) The Security Intention Meeting Series as a way to increase visibility of software security decisions in agile development projects. In: Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES) Canterbury, 26. Aug.–29. Aug, pp 1–8 https://doi.org/10.1145/3339252.3340337
44. Torr P (2005) Demystifying the threat-modeling process. IEEE Secur Priv Mag 3(5):66–70. https://doi.org/10.1109/MSP.2005.119
45. Tuma K, Sion L, Scandariato R, Yskout K (2020) Automating the early detection of security design flaws. In: Proceedings of the 23rd International Conference on Model Driven Engineering Languages and Systems (MODELS) Virtual Event, 16. Oct.–23. Oct, pp 332–342 https://doi.org/10.1145/3365438.3410954
46. van Landuyt D, Joosen W (2020) A descriptive study of assumptions made in LINDDUN privacy threat elicitation. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing (SAC) Brno, 30. Mar.–03. Apr, pp 1280–1287 https://doi.org/10.1145/3341105.3375762
47. vom Brocke J, Simons A, Riemer K, Niehaves B, Plattfaut R, Cleven A (2015) Standing on the Shoulders of Giants: Challenges and Recommendations of Literature Search in Information Systems Research. CAIS. https://doi.org/10.17705/1CAIS.03709
48. Waschke M (2017) What Has the Industry Done? In: Waschke M (ed) Personal Cybersecurity. Apress, Berkeley, pp 175–192
49. Weir C, Becker I, Blair L (2021) A passion for security: intervening to help software developers. In: Proceedings of the 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) Madrid, 25. May–28. May, pp 21–30 https://doi.org/10.1109/ICSE-SEIP52600.2021.00011