sustainability

MDPI

Article

Agile Development of Secure Software for Small and Medium-Sized Enterprises

Anže Mihelič [1,*], Simon Vrhovec [1] and Tomaž Hovelja [2]

[1] Faculty of Criminal Justice and Security, University of Maribor, Kotnikova 8, 1000 Ljubljana, Slovenia
[2] Faculty of Computer and Information Science, University of Ljubljana, Večna Pot 113, 1000 Ljubljana, Slovenia
* Correspondence: anze.mihelic@um.si

Abstract: Although agile methods gained popularity and became globally widespread, developing secure software with agile methods remains a challenge. Method elements (i.e., roles, activities, and artifacts) that aim to increase software security on one hand can reduce the characteristic agility of agile methods on the other. The overall aim of this paper is to provide small- and medium-sized enterprises (SMEs) with the means to improve the sustainability of their software development process in terms of software security despite their limitations, such as low capacity and/or financial resources. Although software engineering literature offers various security elements, there is one key research gap that hinders the ability to provide such means. It remains unclear not only how much individual security elements contribute to software security but also how they impact the agility and costs of software development. To address the gap, we identified security elements found in the literature and evaluated them for their impact on software security, agility, and costs in an international study among practitioners. Finally, we developed a novel lightweight approach for evaluating agile methods from a security perspective. The developed approach can help SMEs to adapt their software development to their needs.

Keywords: secure software development; security engineering; agile; small and medium sized enterprises; software development management; security

Citation: Mihelič, A.; Vrhovec, S.; Hovelja, T. Agile Development of Secure Software for Small and Medium-Sized Enterprises. *Sustainability* **2023**, *15*, 801. https://doi.org/10.3390/su15010801

Academic Editor: Luis Hernández-Callejo

Received: 10 November 2022
Revised: 21 December 2022
Accepted: 29 December 2022
Published: 2 January 2023

1. Introduction

Software development enterprises follow a certain method when developing software, either an established software development method or a custom in-house method even if only informally defined. Following a software development method, i.e., using the right practices, tools, and techniques, makes the process of developing software more manageable and efficient, resulting in greater competitiveness and success [1]. An adequate software development method is therefore crucial for any software development enterprise trying to produce a competitive product in terms of sustainability, costs, quality, and time invested. Today's market demands increasingly frequent software releases. Traditional software development methods tend to be sequential and rigid, which is not well-suited for the demands of the market [2,3]. Modifications of software due to frequent changes in software requirements are time consuming and costly during software development with traditional methods [4]. To overcome the shortcomings of traditional methods, agile software development methods, such as Scrum and eXtreme Programming, emerged [5,6]. Most agile methods lean on software development iterations and increments, self-organizing teams, face-to-face daily communications among team members, and short feedback loops with customers [7,8]. These characteristics are among the key reason why agile methods offer a highly adaptable, efficient, and fast software development process becoming one of the most popular paradigms for software development today [9].

Security is one of the key features of any software according to established standards [10,11]. Software can be considered secure when it is adequately resistant to the

alterations during intentional or unintentional attacks [12]. Agile software development methods, however, seem poorly suited for secure software development [13]. They predominantly focus on functional requirements of the developed software while security, a quality property, is often neglected [14,15]. Various security-oriented software development methods exist following both traditional and agile software development paradigms [16]. The key issue of these mostly traditional methods is the additional overhead they require as it obstructs rapid software development and adaptability to new requirements. It is therefore unclear how to ensure the development of secure software with agile methods without compromising their agility [17].

The overall aim of this paper is to provide small and medium-sized enterprises with the means to improve their software development in terms of software security despite their limitations, such as low capacity and/or financial resources. The specificities of SMEs are additionally pronounced in enterprises practicing the DevOps concept since the (DevOps) teams are traditionally heavily loaded. Such load usually results in software deliverables' degradation, quality decrease, and minimization of the efficiency of the DevOps teams [18]. Furthermore, SMEs are often focused on providing specific solutions for particular needs of their customers, hence they frequently base their software development methods on agile principles [19]. Since such enterprises face highly competitive pressure, their solutions must rely on highly adaptable and cost-efficient frameworks which provide solutions that meet their needs [20].

This paper tries to achieve the overall aim gradually. First, this paper aims to identify security elements found in the literature since the publication of the agile manifesto at the start of this century. We understand a security element as any role, activity, or artifact added to software development methods in the literature to improve the security of the developed software, regardless of its level of abstraction. Even though strong associations between different components (e.g., a specific role can be associated with a specific artifact or activity) can be present, these usually appear in enterprises with more complex and sophisticated software development methodologies (i.e., multinationals). The evaluation of baseline individual impacts of security elements on costs and agility is, however, especially relevant in the context of SMEs, where the limited multiteam environment and significantly lower change resistance from existing power structures often does not require complex and sophisticated methodologies intricately linking several security components [21]. Additionally, roles, for example, in addition to their association with formal activities and artifacts, also introduce a series of informal activities influencing dynamics and efficiency of the development team, thus their synergies with other elements can vary greatly case by case [22].

The published literature does not differentiate between secure software development methods for large enterprises and SMEs. Therefore, the paper does not focus on security elements for SMEs. Rather, it embraces the variety of security elements found in the literature and looks for means to best adapt their use for SMEs. Second, it aims to group and then evaluate the identified security elements according to three key dimensions: provided security, cost efficiency, and retained agility with SMEs in mind. Third, it aims to provide a way for the software development enterprises to take advantage of the evaluation of security elements. The following research questions operationalize the aims of this paper:

- **RQ1:** How much does the individual type of security element impact the software security, costs, and agility of the software development process?
- **RQ2:** How can assessment of provided security, cost efficiency, and retained agility of security elements be useful to small and medium software development enterprises?

To answer these research questions, we first review the literature on secure software development. Since secure software development can be inspired by other security-centric development contexts [23], the literature from other domains, such as the development of safety-critical systems, is also included. Safety-critical systems are those whose failure may result in substantial property, environmental damage, or in some cases in the endangerment or loss of lives [24]. Since the software is an essential part of such systems, the reliability,

confidentiality, and integrity of software used in safety-critical systems are of utmost importance [25]. The review focuses on security elements of software development methods (i.e., activities, roles, and artifacts) compatible with agile software development. The key dimensions of security elements (increased security, cost efficiency, and retained agility) are evaluated in an international survey among secure software development practitioners. Finally, we develop a novel lightweight approach for the evaluation of formal or informal agile methods used in software development enterprises from the security perspective.

2. Literature Review

Several literature surveys and mapping studies on agile software development methods can be found in the literature [16,26–33]. Recently, reviews of state-of-the-art agile secure software development methods [25,34–36] and the DevSecOps framework [37] emerged. Literature surveys on agile secure software development focus on several perspectives, such as security measures in agile projects [36], agile development of safety-critical systems [25], agile requirements for secure software engineering [35], agile security engineering methods [34], and challenges and solutions of agile secure software development methods [5,38]. Although a paper [37] focused on the DevSecOps framework in a multivocal literature survey, it only attempted to consolidate the definition of the framework rather than identify potential security elements that could be included in the framework or existing secure software development methods [37]. Additionally, BSIMM [39] offers an extensive overview of SSD activities, however, some roles and artifacts proposed in the literature are not included. These surveys offer relatively comprehensive insights into the current state-of-the-art of agile secure software development. They address several aspects of secure software development, such as identification of most frequently used methods, security requirement engineering phases, types of solution approaches [35], security activities [34], and secure software development challenges [38].

None of these studies provide an overview of security elements designed for the agile software development process appropriate for small and medium-sized enterprises (SMEs). Additionally, studies show that more than half of proposed software development approaches remain fully theoretical [33]. This lack of empirical evidence raises concerns about their security-enhancing potential, and associated costs and suitability for agile methods. In this section, we provide an overview of existing approaches to security elements which will later be evaluated according to the mentioned dimensions.

Literature offers several approaches to motivating developers and other team members to "think secure," eventually producing a more secure end product. Gamification as an approach for achieving increased motivation and productivity in the software development process was first proposed as Planning Poker in 2002 [40] but has been more thoroughly discussed in the literature since 2014 [41]. Planning Poker is heavily influenced by the Wideband Delphi method, and the rules do not resemble actual Poker rules, except that players (developers and other team members) hide their cards until a designated time. Incorporating security into agile software development by implementing gamification elements was proposed as a variation of Planning Poker as Protection Poker [42] and Threat Poker [43]. However, motivation is not necessarily achieved through gamification. More traditional approaches to security prioritization include business impact analysis [44], continuous aligning of business and security goals, security goals' identification [45] on predominantly strategic levels, and security triage [46], introducing more security controls [47], security intention recap meetings [48], and identifying architectural security requirements [49] on the operational level.

Analyzing and reviewing the code for security issues and security testing are two important and frequently discussed groups of security elements. Security auditing as code analyses and security tests may be performed continuously within the project cycle, such as code analysis [50], analyses performed with automated testing and analysis tools [44,51], security reviews [52], design and code inspections [53], or towards ends of the project cycle, such as penetration testing [54] or other security tests [47,55]. However, different

approaches require respective specialized knowledge within the development team. Hence, several authors propose involvement and incorporation of at least one new subject (role) into the existing methodology. Most commonly, this is a variation of a security guru, namely security engineer [56], security master [57], security-matter expert [58], or security expert [59]. Additionally to these roles, a security developer [44] or a penetration tester [60] may be included in the process. The core purpose of an additional role is to introduce expert knowledge into the development team by identifying the features that have a security risk based on a security principle, documenting risks in the security backlog (if existent), mitigating identified risks, and performing security testing for selected features. A more comprehensive approach, proposed by Baca et al. [54], includes a set of security roles forming a security group. This group should consist of four security roles with different competencies: (1) a security manager who handles non-technical issues, such as ISO certification, legal aspects, etc., (2) a security architect who is responsible for a technical description of the particular features of business use cases, (3) a security master who is responsible for security activities performed during the development process, and (4) a penetration tester who is responsible for final security-related quality assurance. Even though this holistic approach incorporates security by design efficiently, the major drawbacks are time consumption and an increase in cost. Therefore, holistic approaches remain most suitable for safety-critical systems. An additional solution to introducing new knowledge into a development team is the frequently proposed security element, the security training of the development team [55] or even of all stakeholders [61].

Considering that user stories are a driving force of the agile software development process, alteration of product backlog or incorporation of a new security-specific repository, i.e., backlog, is a sensible solution proposal. Several approaches for security-conscious software development can be found in the literature. By adding an additional security-dedicated backlog [56,57], authors suggest that the development team will have a comprehensive overview of potential security issues and security requirements. The additional proposed solution is altering the product backlog by introducing security-related user stories, such as abuser stories [56] and generic security user stories [62] which are security-related user stories: a set of possible scenarios and threats to the end product. The main difference between abuser stories and generic security user stories is in user story development process. Abuser stories are usually developed in collaboration with all participating parties, whereas generic security user stories are ready-made for development teams to use. In contrast to expanding an existing product backlog, it was also proposed that misuse or abuse cases [63] as an independent element should be incorporated into the development process. This can help developers see the software from the attackers' perspective. A sensible approach to enhance the security-conscious software development is an incorporation of security tags within security repositories and user stories. Security tags may be included in the form of S-Tags and S-Marks as suggested by [8] or as security keywords [52].

Building on the premise that a significant drawback of agile methodologies is a lack of a complete overview of security issues [54], various forms of security planning and modeling with security prioritization and monitoring activities and risk management activities are proposed. The main difference is that planning, modeling, prioritization, and monitoring activities are commonly performed at the operational level. These are activities such as security "games" (as mentioned earlier), security test cases and test plan reviews, attack surface recognition and reduction [44], security goals identification [45,51], security meetings [13], defining criticalities [44], and prioritization of security requirements [64]. On the other hand, risk management activities are performed at the strategic level. Traditional activities associated with risk management, such as risk identification, risk analysis, risk assessment, and their variations, are proposed in the literature [50,54,55,65,66]. Various reports on security matters (either security issues or audit reports) commonly accompany security activities [44].

Security elements discussed in this section are grouped and summarized in Table 1. The table includes the code to which the results section refers, a consolidated name of the

group of security elements, a brief description, and sources in which particular security elements can be found in the literature.

Table 1. Overview of security element groups.

Security Element Group	Description	Sources
Roles		
Security guru (*SGuru*)	A security guru (also known as *security manager, security master*) is respected for his security knowledge and leads the development process from a security perspective on the strategic level.	[53,54,67,68]
Security developer (*SDeveloper*)	A security developer is not a programming role but a role that designs security tests, risk analyses, threat models, and attack surface analysis.	[44,54]
Penetration tester (*PenTester*)	A penetration tester tests for known vulnerabilities and attack vectors and tries to find possible exploits in developed software.	[54]
Security team (*STeam*)	A security team is a group of several security-related roles (e.g., *penetration tester, security manager, security architect*).	[50,54,67]
Activities		
Security auditing (*SAudit*)	Security auditing includes various audits and reviews that help assess the end product's overall security posture (e.g., *security review, code review, test case code review, security auditing*).	[44,50,52,53]
Security analysis and testing (*SAnalyTest*)	The security analysis and testing group is a set of activities for analyzing and testing software for their security performance (e.g., *vulnerability analysis, penetration testing, code inspection, risk-based security tests*).	[8,44,45,47,49–53,55,67–69]
Security training (*STrain*)	Security training is an activity aimed to increase security-related knowledge among developers and other stakeholders in the software development activity.	[44,55,61]
Security prioritization and monitoring (*SPriorMoni*)	The security prioritization and monitoring group is a set of activities that prioritize specific security goals, requirements, activities, and monitor their implementation (e.g., *business impact analysis, protection poker, continuous aligning of business and security goals*).	[42–46,48,51,56,64,70,71]
Risk management (*RiskManag*)	Risk management activities help identify, assess, and control threats to the end product on a strategic level (e.g., *risk identification, risk assessment, hazard analysis*).	[45,47,50–55,64–66]
Security planning and threat modeling (*SPlanModel*)	Security planning and threat modeling are activities for structured identification of potential operational threats, such as structural vulnerabilities or the absence of appropriate safeguards (e.g., *security modeling, threat modeling*).	[44,45,51,52,55,67,68]
Security requirements engineering (*SReqEng*)	Security requirements engineering activities focus on defining, documenting, and maintaining security requirements in the engineering design activity (e.g., *defining security requirements, formulation of abuser stories, defining security-related user stories*).	[49,50,52,55,56,69]
Artifacts		
Security requirement artifacts (*SReq*)	Security requirement artifacts are backlog-related elements that put a focus on security in the process of requirement acquisition (e.g., *security user stories, abuser stories misuse cases, abuse cases*).	[45,49,51,55,56,62]
Security repositories (*SReposi*)	Security repositories help manage security risks by providing a requirement check list for increased end-product security (e.g., *security backlog, security requirements repository, safety product backlog*).	[57,64–66,72]

Table 1. *Cont.*

Security Element Group	Description	Sources
Security reports (*SReports*)	Security reports are documents written after a specific security activity is performed (e.g., *security testing report, test phase code review report, security mechanism review report, security audit report*).	[44]
Security tags (*STags*)	Security tags help developers to be aware of the security relevance of user stories. They identify parts of the emerging software that need security verification (e.g., *S-Tags, S-Marks, security keywords*).	[8,52]
Security policies (*SPolicies*)	Security policies are documents that help developers planning secure software development by providing security-related standards and solutions (e.g., *security test plan, secure coding policies, security related coding standards*).	[44,50,52,55,56,61]

Even though several approaches can be found in the literature proposing a relatively large amount of security elements, only a fraction (approximately one-fifth) of those approaches were tested in industrial settings [33]. Hence, little is known about their direct impact on agility and costs. In rare cases, studies report on cost increases [54] and a possible compromise of agility of some elements [45,51]. However, to the best of our knowledge, there is no systematic evaluation of all security elements or their corresponding security element groups regarding their impact on the agility of the software development process and costs of their implementation. Therefore, this paper aims to fill this gap by conducting an evaluation of security element groups from the practitioners' perspective.

3. Materials and Methods

To answer the first research question, we evaluated the impacts of security element groups on security, costs, and agility of software development. We conducted an international survey among secure software development practitioners between April and May 2022 on the *prolific.co* platform in two phases. In the first phase, we conducted a screening survey under the criteria "Industry: Software." No other limitations were applied. The screening survey included a definition of secure software engineering and questions on their work experience with software development and agile software development (in years) and their work experience with secure software engineering (in years). We received 1053 responses in total, out of which 167 respondents had three or more years of experience with secure software engineering. These were considered as appropriately knowledgeable to participate in the second phase of the survey.

In the second phase, we invited the 167 respondents with three or more years of experience with secure software engineering to participate in a survey to evaluate security elements. The questionnaire in the second phase was developed to measure three key performance dimensions of each security element: (1) *provided security* to developed software (i.e., the resistance of software to attacks), (2) *increased costs* of software development (e.g., man-hours, direct costs), and (3) *compromised agility* of the software development solution (i.e., a departure from the agile principles). The survey questionnaire first included a short introduction with an explanation of the evaluation criteria presented above. The main part of the questionnaire consisted of three sections, one for each element group (i.e., roles, activities, and artifacts). A description was provided for each security element group, followed by the question: *How would implementing an average element from this cluster affect an agile method without security-related elements?* The three key performance dimensions were evaluated on 7-point unipolar scales: from 1 (*no effect on security*) to 7 (*considerably improve security*) (provided security), from 1 (*no effect on costs*) to 7 (*considerably increase costs*) (increased costs), and from 1 (*no effect on agility*) to 7 (*considerably reduce agility*) (compromised agility). All questionnaire items were self-developed.

Out of the 167 invited participants, we received 117 fully and 13 partially completed surveys. After the data cleaning, 112 responses were appropriate for further analyses,

returning a response rate of 62.3 percent. Average response duration was 6 min and 46 s. The respondents' age ranged from 22 to 62, with a mean of 36.2 years (SD = 9.2). Respondents' experience in secure software engineering ranged from 3 to 20 years, with a mean of 6.0 years (SD = 3.8). Other demographic characteristics of the respondents are presented in Table 2.

Table 2. Demographic characteristics of the sample.

Characteristic		Number	Share
Gender	Female	27	24.0%
	Male	85	76.0%
Primary role	Software developer	73	65.2%
	Project manager	24	21.4%
	Security expert	4	3.6%
	Other	10	8.9%
Formal education	High school or less	17	15.2%
	Bachelor's degree or equivalent	55	49.1%
	Master's degree or equivalent	39	34.8%
	Doctoral degree or equivalent	1	0.9%
Country	Australia	1	0.9%
	Belgium	1	0.9%
	Canada	6	5.4%
	Chile	1	0.9%
	Denmark	1	0.9%
	France	2	1.8%
	Germany	3	2.7%
	Greece	1	0.9%
	Ireland	2	1.8%
	Israel	1	0.9%
	Italy	6	5.6%
	Mexico	4	3.6%
	Netherlands	1	0.9%
	Poland	9	8.0%
	Portugal	7	6.3%
	South Africa	6	5.4%
	Spain	7	6.3%
	Sweden	2	1.8%
	United Kingdom	31	27.7%
	United States	20	17.9%

4. Results

To facilitate the interpretation of the results, the scores for *increased costs* and *compromised agility* were reversed (i.e., recoded from 1, 2, . . . , 7 to 7, 6, . . . , 1, respectively). These performance dimensions were also renamed to *cost efficiency* and *retained agility*, respectively, to reflect the change. For all three key performance dimensions, a lower score means worse performance (i.e., less provided security, lesser cost efficiency, and less retained agility) and a higher score means better performance (i.e., more provided security, better cost efficiency, and more retained agility). A summary of overall means for roles, activities, and artifacts is presented in Table 3.

Table 3. Mean values for all roles, activities, and artifacts according to all three evaluation dimensions.

	Roles	Activities	Artifacts
retained agility	4.27	4.40	4.64
cost efficiency	3.27	3.76	5.52
security	5.70	5.26	4.64

In the following, we present means, standard deviations, medians, and modes for the three key performance dimensions of all security element groups. Security elements are grouped together into a table for each type. They are sorted descending according to provided security.

The evaluation of roles is presented in Table 4. The *Security team* provides most security not only among roles but among all security element groups. However, it is also the least cost-efficient and retains the least agility. Although a *Security guru* is among the most frequently proposed security elements in the literature, a *Penetration tester* or *Security developer* may be a more cost-efficient alternative providing a similar or higher level of security and retaining a comparable level of agility. Summarily, roles provide the most security compared to activities and artifacts ($M_{security}$ = 5.70). Roles are, however, the least cost-efficient of the three (M_{costs} = 3.27) and retain the least agility ($M_{agility}$ = 4.27). The univariate skewness and kurtosis values for the roles ranged from −1.77 to 0.67 and −0.96 to 2.93, respectively, indicating an approximately normal data distribution [73].

Table 4. Assessment of roles.

Evaluation Criteria	Mean	SD	Median	Mode
Security team				
retained agility	3.94	1.60	4	3
cost efficiency	2.80	1.50	3	3
security	6.05	1.37	7	7
Penetration tester				
retained agility	4.32	1.66	4	6
cost efficiency	3.42	1.45	3	3
security	5.64	1.33	6	7
Security guru				
retained agility	4.47	1.52	5	5
cost efficiency	3.38	1.54	3	3
security	5.56	1.37	6	7
Security developer				
retained agility	4.36	1.59	4	4
cost efficiency	3.47	1.41	3	3
security	5.54	1.42	6	6

Table 5 presents the evaluation of activities. *Security analysis and testing* provides the most security. However, it is also the least cost-efficient and retains the average agility among the activities. *Security training* retains the most agility and provides the fourth-most security while being averagely cost-efficient. Even though *Security planning and threat modeling* and *Risk management* are among those that are most frequently found in the literature, our results indicate that their provided security is among the lowest among the activities. In summary, activities seem to be between roles and artifacts regarding provided security, retained agility, and cost efficiency ($M_{security}$ = 5.26, M_{costs} = 3.76, $M_{agility}$ = 4.40). The univariate skewness and kurtosis values for the activities ranged from −1.23 to 0.29 and −1.04 to 1.87, respectively, indicating an approximately normal data distribution [74].

Table 5. Assessment of activities.

Evaluation Criteria	Mean	SD	Median	Mode
Security analysis and testing				
retained agility	4.28	1.53	5	5
cost efficiency	3.63	1.40	3	3
security	5.52	1.25	6	6
Security prioritization and monitoring				
retained agility	4.13	1.72	4	3
cost efficiency	3.69	1.52	4	3
security	5.29	1.39	5	5
Security auditing				
retained agility	4.06	1.76	4	5
cost efficiency	3.54	1.47	3	3
security	5.24	1.53	6	6
Security training				
retained agility	4.88	1.85	5	6
cost efficiency	3.65	1.47	3	3
security	5.19	1.35	5	6
Security requirements engineering				
retained agility	4.43	1.64	4	4
cost efficiency	3.71	1.39	4	3
security	5.13	1.38	5	6
Security planning and threat modeling				
retained agility	4.43	1.46	4	4
cost efficiency	3.98	1.41	4	4
security	5.08	1.32	5	6
Risk management				
retained agility	4.55	1.67	5	6
cost efficiency	4.07	1.51	4	3
security	4.81	1.57	5	5

The evaluation of artifacts is presented in Table 6. The only artifacts with a mean value greater than 5 are *Security policies*. Besides providing the most security among artifacts, it retains the least agility but is the second most cost-efficient artifact. *Security reports* together with *Security tags* provide the least security among all security element groups regardless of their type. Overall, artifacts are the most cost-efficient (M_{costs} = 5.52) and retain the most agility ($M_{agility}$ = 4.64). However, they also provide the least security ($M_{security}$ = 4.64). The univariate skewness and kurtosis values for the artifacts ranged from −0.67 to −0.13 and −1.03 to −0.13, respectively, indicating an approximately normal data distribution [74].

To identify possible outliers, we present the results in the boxplots in Figures 1–3 for roles, activities, and artifacts, respectively. As seen from the figures, only four definite outliers in all three boxplots can be observed. The outliers are presented in the security dimension of two elements (security team, security analysis and testing) that already score very high in security. Thus, removing them would increase their security scores even more and would not impact the relative position of elements in scatter plot quadrants.

Table 6. Assessment of artifacts.

Evaluation Criteria	Mean	SD	Median	Mode
Security policies				
retained agility	4.46	1.91	5	6
cost efficiency	4.54	1.68	5	5
security	5.07	1.42	5	6
Security requirement artifacts				
retained agility	4.56	1.69	5	5
cost efficiency	4.46	1.52	4	4
security	4.79	1.44	5	5
Security repositories				
retained agility	4.58	1.65	4	4
cost efficiency	4.41	1.56	5	5
security	4.68	1.53	5	5
Security reports				
retained agility	4.62	1.69	5	4
cost efficiency	3.35	1.57	4	5
security	4.34	1.51	4	5
Security tags				
retained agility	4.97	1.69	5	6
cost efficiency	4.85	1.77	5	7
Security	4.29	1.62	4	6

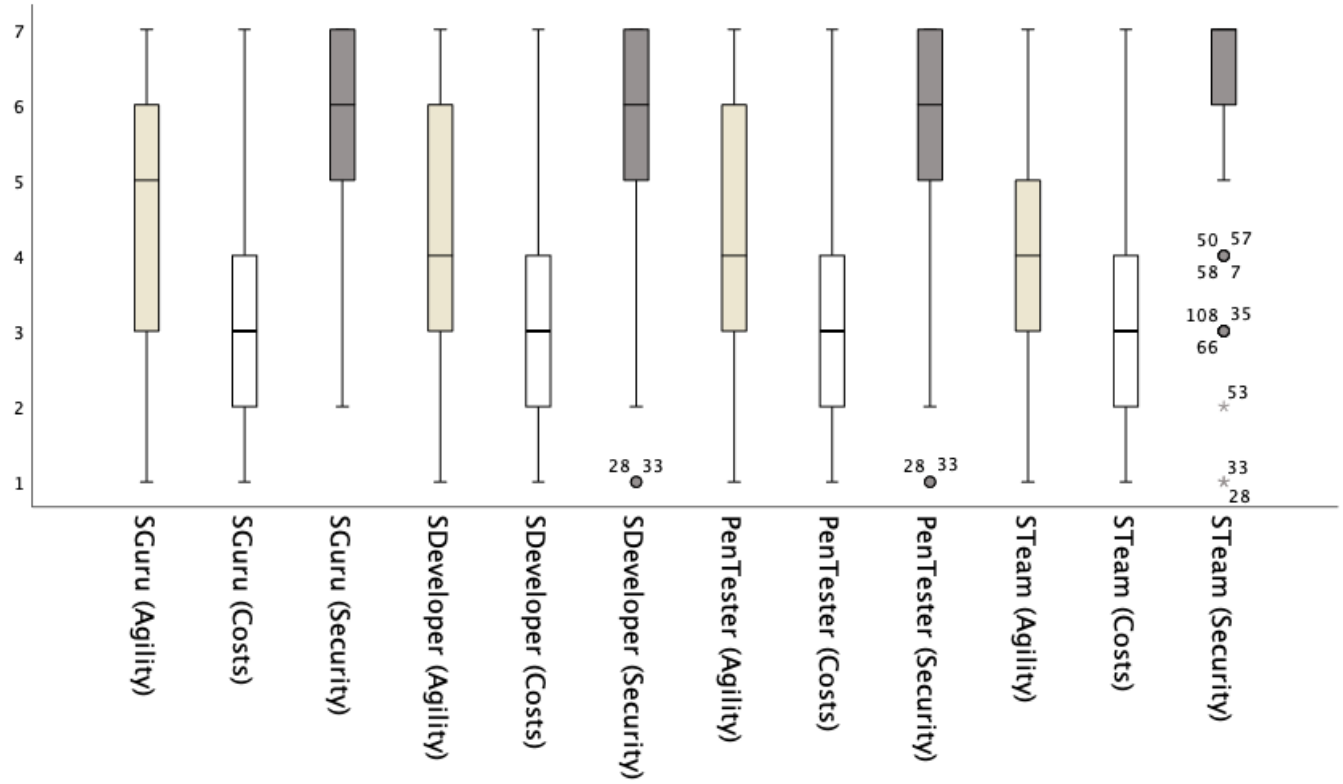

Figure 1. Results graphically presented in **a** boxplot diagram for roles.

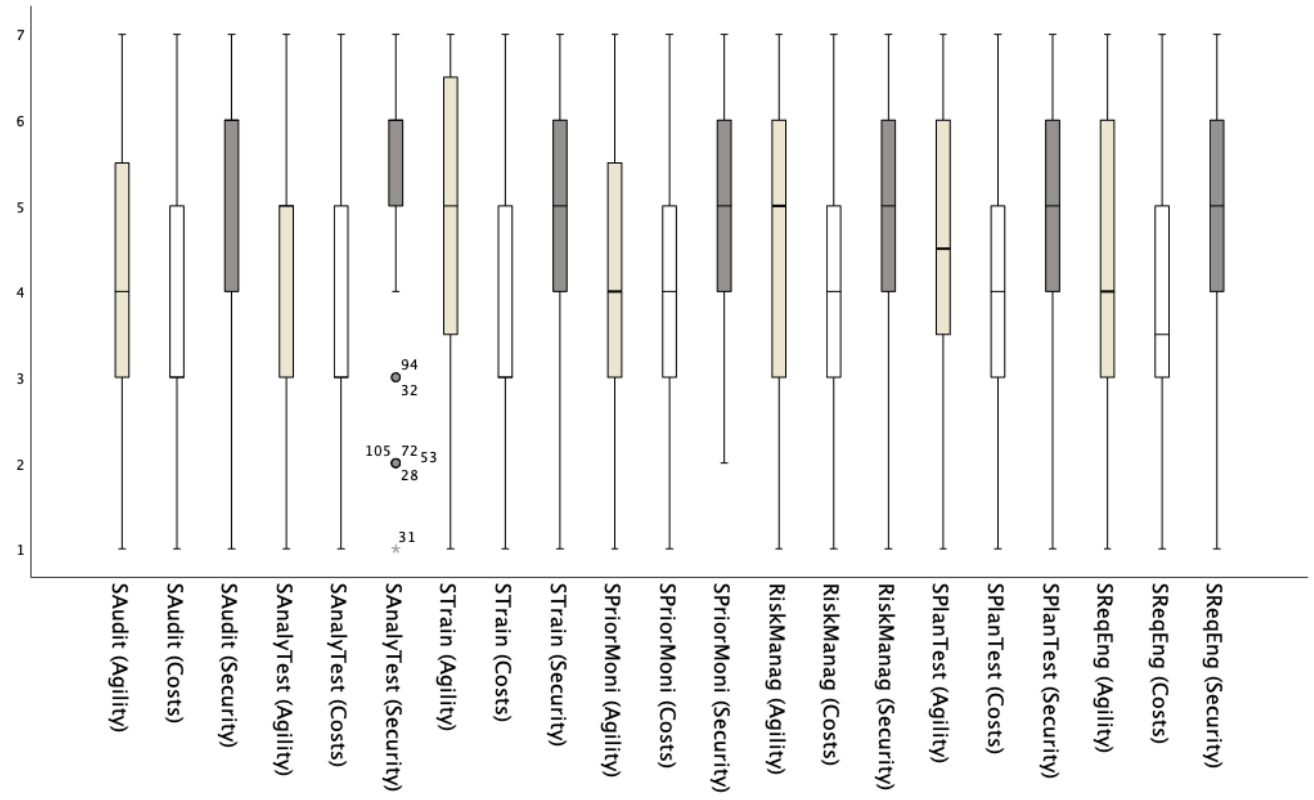

Figure 2. Results graphically presented in a boxplot diagram for activities.

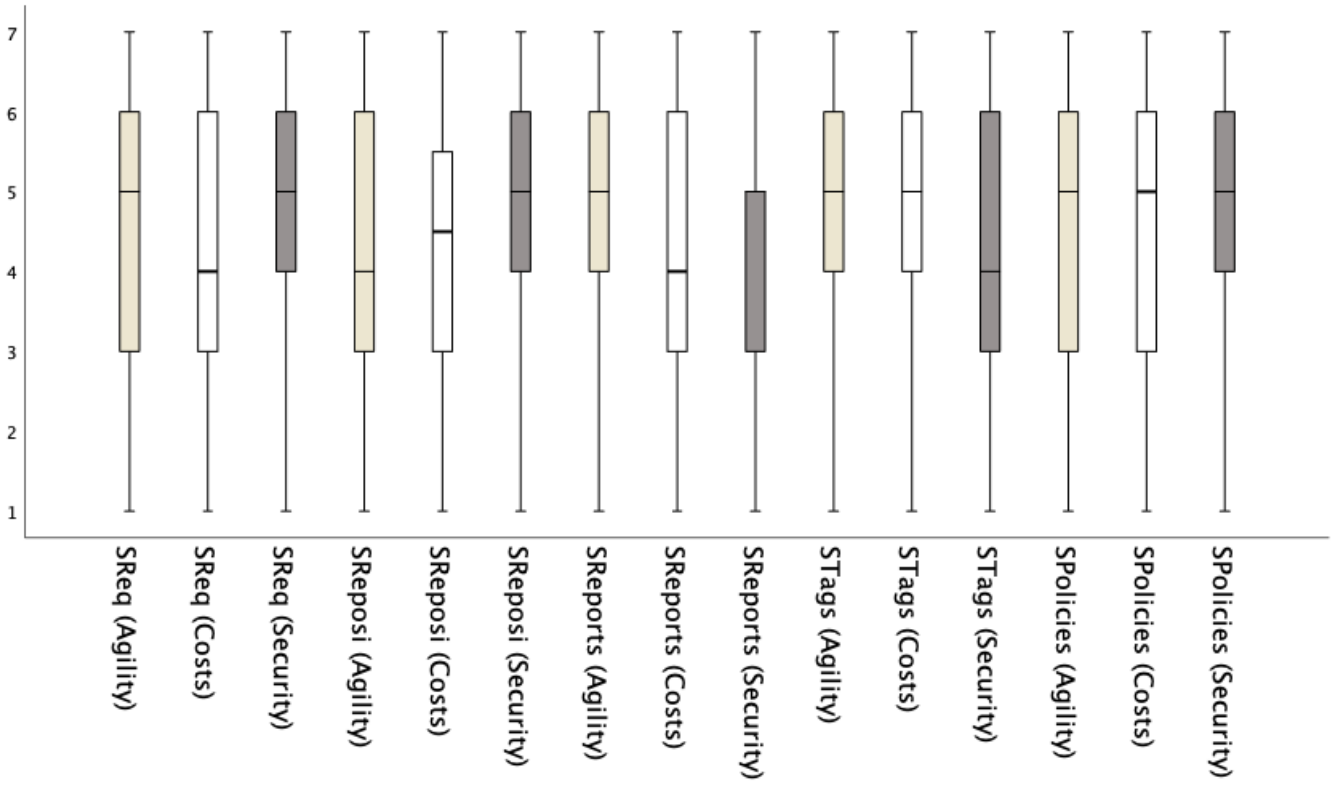

Figure 3. Results graphically presented in a boxplot diagram for activities.

5. Proposed Model for SMEs

To answer the second research question, we are proposing a lightweight approach for the evaluation of secure software development method elements. The proposed approach can be considered lightweight since it has few rules and practices and is simple and easy to learn. Further, the improvements achieved through the implementation of the approach can be applied to different development projects later. It is a separate activity that can be used as a tool or a first steppingstone when planning changes in development methods from a security perspective or implementing any novelty where compromised agility, cost efficiency, and added security should be considered. The approach consists of three phases: (1) identification of security elements comprising the software development method used by a software development enterprise, (2) evaluation of identified security elements, and (3) improvement of the existing method (see Figure 4).

Figure 4. The software development method evaluation approach.

The *first phase* aims to gain insights into the existing software development method needed for evaluating and improving it. This phase enables software development enterprises to use the proposed approach for *any* software development method even if there is no formal method or if software development is "as agile as it gets" (i.e., denial of the existence of any method). In this phase, enterprises first identify *all* elements comprising their software development method, whether they are focused on security or not. By identifying all elements, enterprises will get an overview of the elements and the relationships between them. An in-depth insight into the existing software development method is necessary to avoid unwanted effects of potential improvements, such as the introduction of elements duplicating parts of existing elements or conflicting with them. Next, security elements need to be identified. This enables the evaluation of security elements that are already in place.

The *second phase* aims to evaluate the security elements in place. The results of our international evaluation study provide the grounds for such evaluation. In this phase, software development enterprises need to determine which security elements in place can be mapped to which security element groups (as identified in this Section 2). Since the security element groups cover nearly all security elements found in the literature, most if not all security elements in place can be mapped to one of the security element groups. This enables enterprises to seamlessly evaluate the provided security, cost efficiency, and retained agility of the security elements in place. If enterprises, however, identify a security element in place that cannot be mapped to any of the security element groups, the evaluation needs to be repeated among available experts for *all* security elements in place to avoid skewed results due to potential differences in the respondents' bias of our study and the new evaluation.

The *third phase* aims to provide recommendations for improving the existing software development method. The existing software development method can be improved in various ways. For example, it can be more security-focused, cost efficient, and/or agile. To facilitate decision making, the evaluation of security elements is visualized with scatter

plots. One scatter plot shows the trade-off between provided security and cost efficiency, and another one shows the trade-off between provided security and retained agility. Different quadrants in the scatter plots provide their own heuristic recommendations for improving the existing software development method through security elements placed in it. The quadrants were named as shown in Figure 5 to ease the understanding of the meaning of security element placements: *Goldilocks*, *Possible overkill*, *Prospects*, and *Wastrels*.

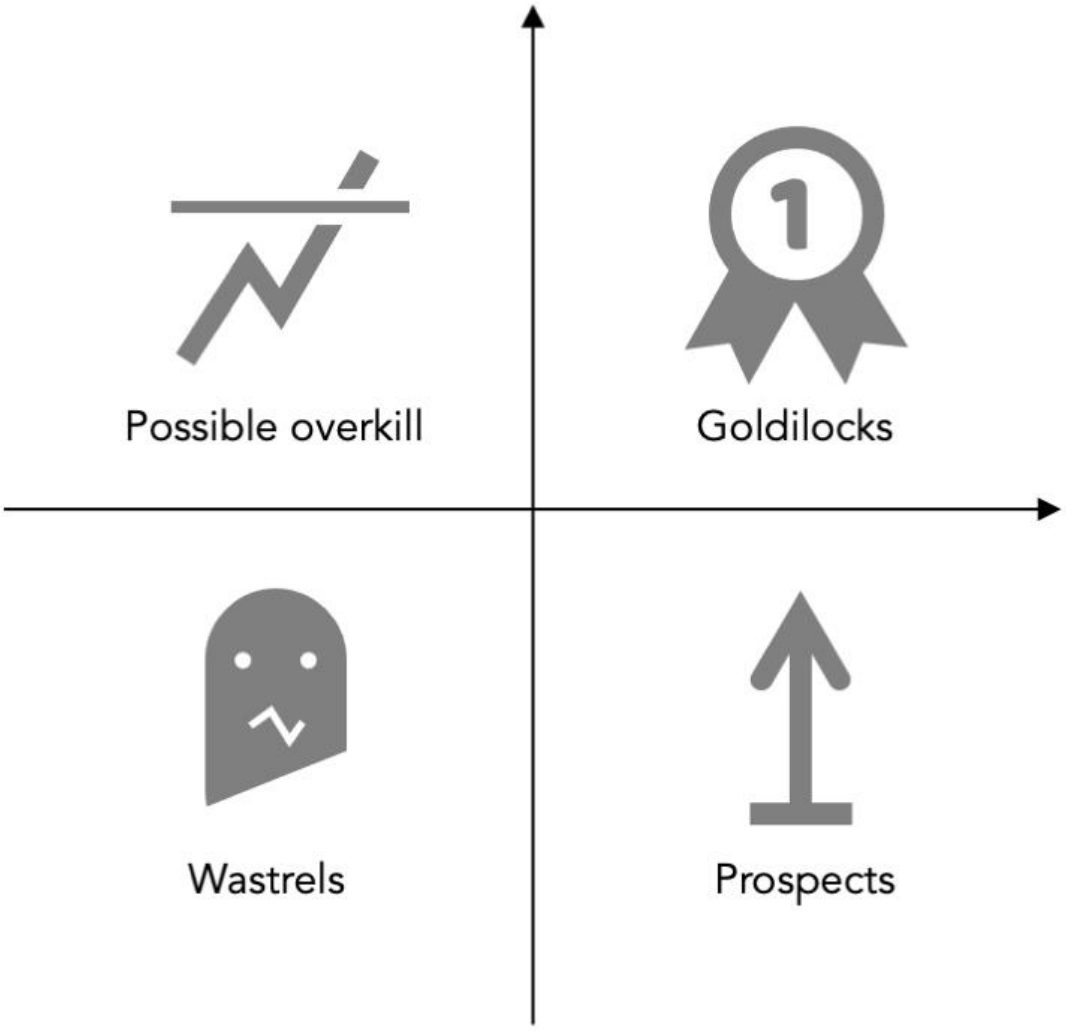

Figure 5. An example of categorization of security elements in the scatter plots.

The *Goldilocks* quadrant consists of security elements with above-average provided security and above-average cost efficiency or retained agility. Security elements in this quadrant have the best ratio between provided security and cost efficiency or retained agility and can be thus regarded as the most suitable for small and medium software development enterprises that cannot afford less cost-efficient security elements or want to lose too much agility in their software development. The *Possible overkill* quadrant contains security elements with above-average provided security but below-average cost efficiency or retained agility. Security elements in this quadrant are a potential security overkill for small and medium software development enterprises. Even though they provide above-average security, they may be too costly or compromise too much agility. Nevertheless, they may be suitable for developing safety-critical software and other software when security is a top priority (e.g., online banking). The *Prospects* quadrant consists of security elements with below-average provided security and above-average cost efficiency or retained agility. Security elements in this quadrant can underperform in terms of provided security. If security is not a top priority, these security elements can be considered as a cost-efficient

way to provide some security while retaining agility. For small and medium enterprises, they can also be attractive as alternatives for less cost-efficient security elements providing more security, especially coupled with some such elements. The *Wastrels* quadrant contains security elements with below-average provided security and below-average cost efficiency or retained agility. Since these security elements are less cost efficient or retain less agility while providing less security, there are poor grounds for including them in software development methods. Unless they have some other added value justifying their inclusion, these security elements may be dropped. An analysis of the impact on security may be needed before dropping any security elements. If needed, the dropped security elements may be replaced by other security elements.

To demonstrate the use of scatter plots, we visualized the results of the international survey among published secure software development authors. The *x*-axis is placed at the provided security median for all security element groups regardless of their type. The *y*-axis is placed at the median of cost efficiency (Figure 6) and retained agility (Figure 7) for all security element groups regardless of their type.

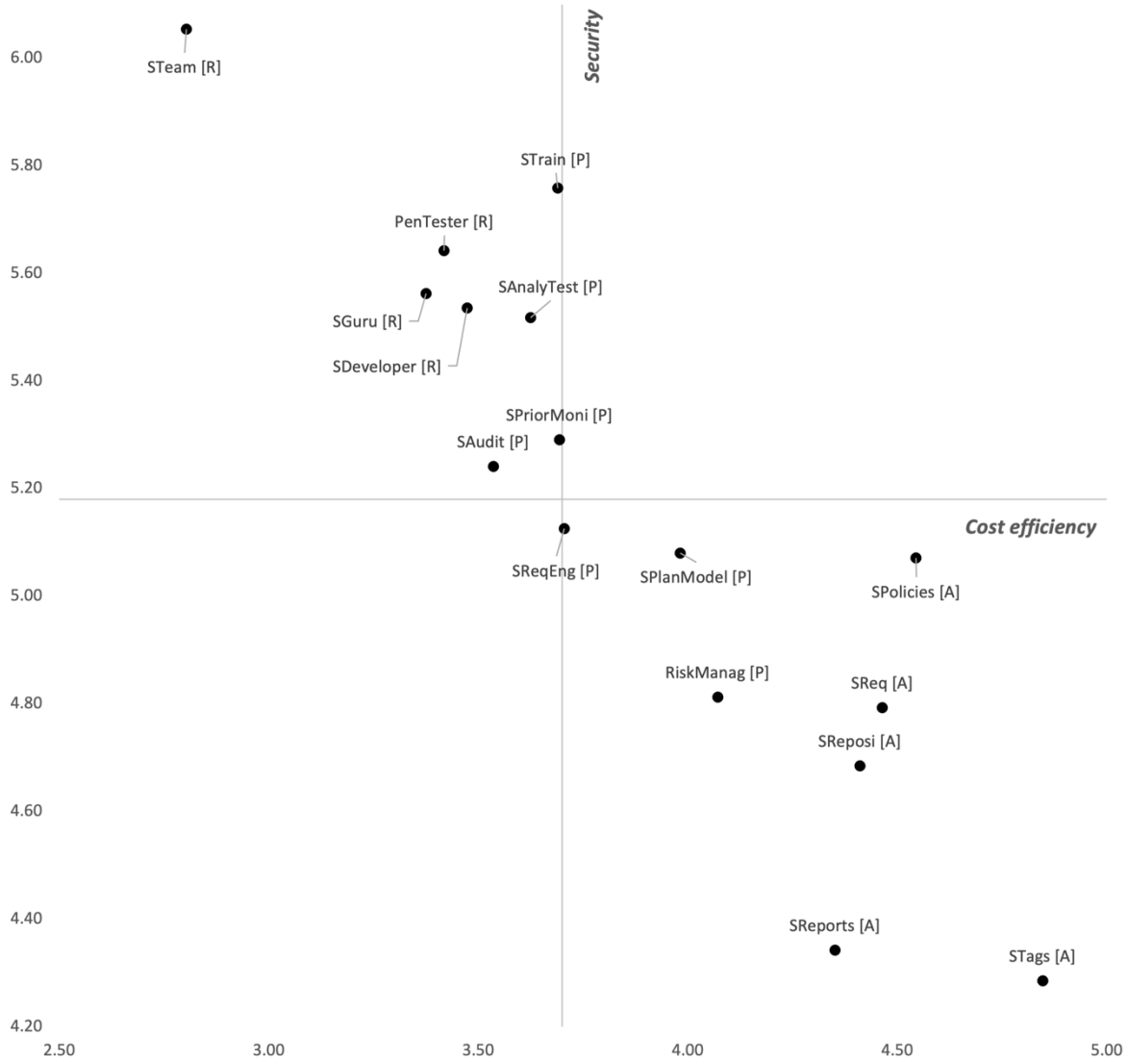

Figure 6. Scatter plot of security element groups showing their provided security and cost efficiency (higher value means better cost efficiency and more provided security).

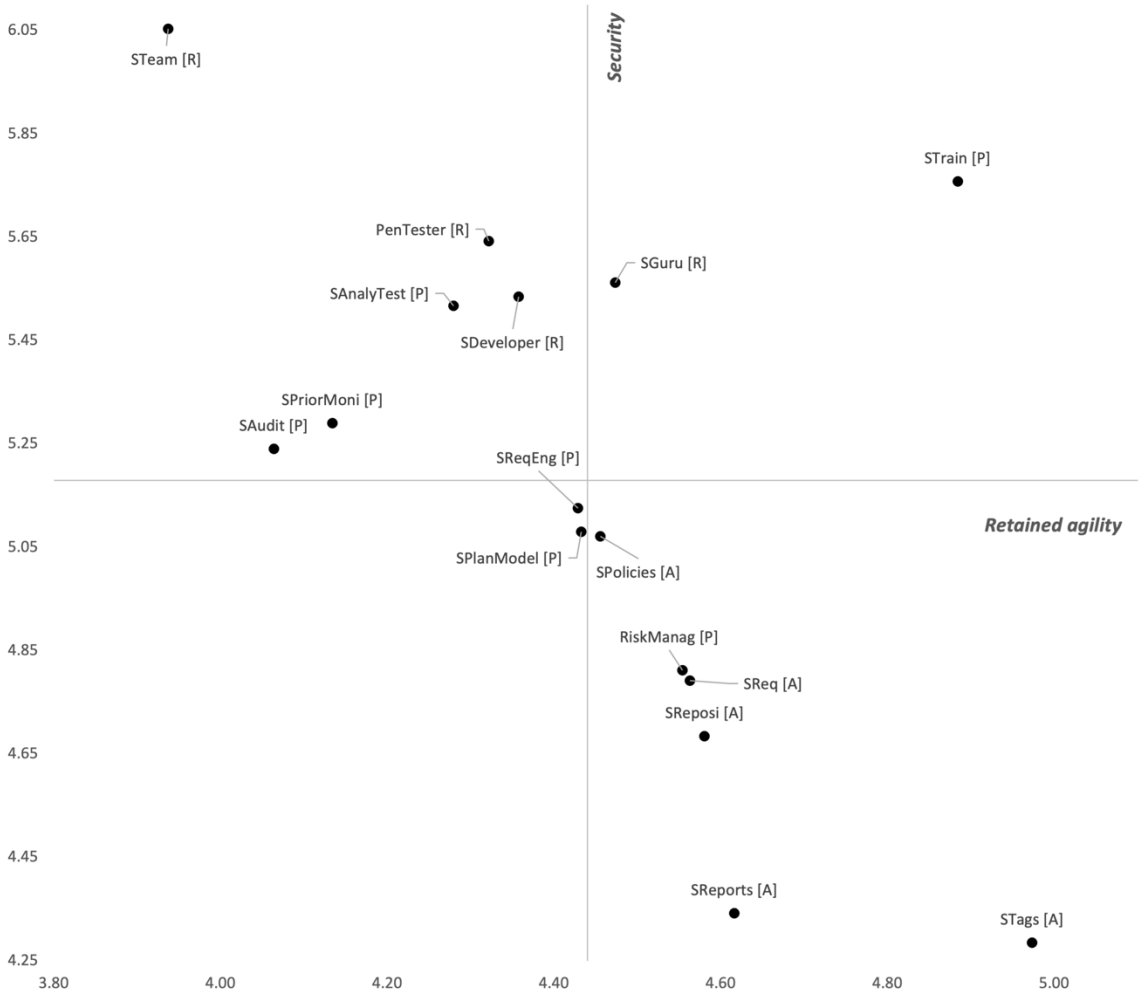

Figure 7. Scatter plot of security element *groups* showing their provided security and retained agility (higher value means more retained agility and more provided security).

The results seen in Figure 3 indicate a negative relationship between security and costs, as less cost-efficient security elements provide more security. A similar negative relationship can be seen in in Figure 4, whereas less agile-threatening security elements provide more security. Only very few outliers can be observed.

No security element groups were placed in the *Goldilocks* quadrant in both scatter plots. However, *Security training* is placed in the Goldilocks quadrant in the security–retained agility scatter plot, while being reasonably close to this quadrant in the security–cost efficiency scatter plot. Hence, this security element may be the most appropriate for small and medium software development enterprises. *Security guru* was placed in the *Goldilocks* quadrant in one scatter plot but in the *Possible overkill* quadrant in the other, while the *Security prioritization and monitoring* group was, in both cases, placed in the *Possible overkill* quadrant; however, in the security–cost efficiency scatter plot, it was placed reasonably close to the *Goldilocks* quadrant. Hence, the *Security guru* is not as cost-efficient, but it does provide above-average security while not interfering with the agility of the method. Since the *Security prioritization and monitoring* group provides a reasonable cost efficiency while offering above-average security, it could be considered in the same context as the *Security guru.* Small and medium enterprises may consider the trade-off between costs and agility when deciding among these security elements. Six security element groups were placed in the *Prospects* quadrant in both scatter plots: *Security policies*, *Risk management*, *Security requirements artifacts*, *Security repositories*, *Security reports*, and *Security tags.* Although these element groups provide below-average security for the end product, they may still be

considered when security is essential but not the highest priority or in cases where an enterprise plans to invest fewer resources or sacrifice agility as little as possible but still aims to introduce secure software development elements into their method. The *Wastrels* quadrant seems to be inhibited similarly to the *Goldilocks*. No security element groups were placed in this quadrant in both scatter plots. Only *Security requirements engineering* and *Security planning and modeling* were placed in this quadrant in the security–retained agility scatter plot. None of the security element groups were placed within *Wastrels* in the security–cost efficiency scatter plot. These two security elements should be considered for incorporation into the development method of an SME when there is a reasonable justification for their inclusion, mainly since they introduce an approximately average level of security in comparison to others and may affect the agility of the process.

To provide an informative overview of the placements of security element groups into quadrants, we summarize the findings of our study in Table 7.

Table 7. Placement of security element *groups* into quadrants.

Security Element	Costs	Agility
Roles		
Security guru	Possible overkill	Goldilocks
Security developer	Possible overkill	Possible overkill
Penetration tester	Possible overkill	Possible overkill
Security team	Possible overkill	Possible overkill
Activities		
Security training	Possible overkill	Goldilocks
Security prioritization and monitoring	Possible overkill	Possible overkill
Security auditing	Possible overkill	Possible overkill
Security analysis and testing	Possible overkill	Possible overkill
Risk management	Prospects	Prospects
Security requirements engineering	Prospects	Wastrels
Security planning and threat modeling	Prospects	Wastrels
Artifacts		
Security requirement artifacts	Prospects	Prospects
Security repositories	Prospects	Prospects
Security tags	Prospects	Prospects
Security policies	Prospects	Prospects
Security reports	Prospects	Prospects

6. Discussion

This paper aimed to provide SMEs with the means to improve their software development in terms of software security. To achieve this, it centers around the concept of security elements—those elements (i.e., activities, roles, artifacts) of software development methods [73] that provide security. Security elements were first collected from published literature and then grouped. The security element groups resemble the core phases of secure software development suggested by [16]: software requirements security, software design security, software construction security, and software testing security. This paper, however, breaks down these core phases into smaller parts according to the type of method element. Several previously proposed secure software development frameworks, approaches, and maturity models, e.g., [39,75,76], provide some support for the security element groups presented in this paper.

According to the opinion of the surveyed practitioners, *roles* provide the most security among different types of security element groups. All roles have above-average provided security indicating that security expertise is one of the key factors in ensuring the security of developed software. The *security team* typically consists of several security specialists whose expertise highly impacts the security of the developed software. However, it also compromises the agility of the software development process (e.g., by introducing various security controls into existing activities) and significantly increases its costs. It is widely and commonly known that security specialists, such as *penetration testers* and *security gurus*, are highly sought after and relatively expensive. Such findings are in line with the current literature reporting up to a 500 percent increase in costs due to the inclusion of several security roles in the development process [54]. According to the survey results, the *security developer* is the most balanced role providing considerable security at relatively low costs while also retaining agility. Additionally, *security developers* are actively involved in the development process and thus importantly improve the expertise of the development team, which may be complemented by the *penetration tester* (involved during testing or later) and *security guru* (not actively involved in the development process). Roles seem to compromise agility the most, which can be associated with the prerequisite of more sequential organization of the development process—e.g., stages at which those roles are most frequently used at the development stages: analysis, design, coding, and testing [45].

As characteristic for agile methods [73], *activities* are the most represented type of security elements both in terms of quantity and diversity. They retain the second least agility among all three types of security element groups, which may be due the fact that including new activities into agile methods inherently compromises agility to an extent. The effects on agility and costs can be especially pronounced if the development team does not have the necessary knowledge to efficiently perform the introduced activities. *Security training* is the only activity in the *Goldilocks* quadrant for agility and is placed extremely close to it in terms of cost efficiency. This is especially valuable since its results increase the security knowledge of the development team, directly affecting one of the key factors in providing security. Not only does *security training* not compromise the agility of the software development process in the long term but also the costs invested are subsidized across different projects as the acquired security knowledge stays with the development team members. Additionally, security training is typically held outside of the development process, hence it does not affect its agility. This finding is also consistent with the published literature emphasizing the importance of security skills and training [68,77]. *Risk management* is among the more popular security elements for agile methods. Our results, however, suggest that it provides the least security among activities, which may be attributed to its focus on the strategic level of security planning and a lacking practical application during project implementation. This can be also seen in some degree of compromised agility if this activity is implemented in practice. Consistent with these findings, a study focusing on secure software development practices indicated that these practices could be characterized as compliance-based, arbitrary, late, and error-driven rather than risk-oriented [78].

Although *artifacts* are generally the most cost-efficient and retain the most agility, they perform relatively poorly in providing security. For example, *security reports* are among the most widespread security elements, yet they score very low on the security scale. Not all artifacts perform poorly, though. For example, *security policies* fall into the upper part of the *Prospects* quadrant for both costs and agility. *Security requirement artifacts* and *Security repositories* do not fall far behind. These artifacts guide the development team to think about the developed software more from the security perspective, potentially somewhat making up for lacking security expertise of developers.

6.1. Theoretical Implications

Our research advances the current literature in several ways. First, we have developed a novel approach for multi-dimensional evaluation (i.e., security, cost, agility) of agile secure software development methods. Our approach focuses on the evaluation of security

elements. It is designed in a way that enables evaluation of any agile method since even the least defined methods consist of certain recurring elements (i.e., artifacts, roles, activities) that can be evaluated. The evaluation of multiple dimensions of security elements provides insights from different viewpoints, which enables a more comprehensive evaluation of software development methods from the security perspective.

Second, the results of the international study among practitioners indicate that there may be a negative relationship between provided security of the developed software and retained agility of the development process and in the relationship between provided security and cost efficiency. Future work may focus on identifying prospective security elements beyond those already known to be applicable to agile methods to confirm such a finding.

Third, our study provides a list of 16 security element groups. This list can be used as a baseline for scholars and practitioners developing new security elements or combining them into holistic secure software development methods. The provided evaluation of security elements may help them improve their decision-making process when selecting the security elements that provide the best combination of a desired level of security and acceptable levels of costs and agility. The list of security element groups may be useful beyond the current timeframe in which it was created. The list is based on a review of two decades of agile secure software development research; therefore, it may not be too surprising if future security elements can be consolidated into one of those already on the list.

Fourth, the evaluation of security elements appears to point to an interesting finding. Several relatively cost-efficient security elements that provide ample security while retaining agility (i.e., *security training*, *security requirement artifacts*, and *security developer*) deal with increasing the security expertise of software developers, thus improving the whole software development process. Future work may thus focus on developing new long-term approaches for incrementally improving the security expertise in the development team, which would not significantly compromise the agility or increase the costs of software development. However, empirical evidence would be very needed to determine the actual effectiveness of such approaches in the long run.

6.2. Practical Implications

This paper also has several practical implications. First, the proposed approach offers a practical and cost-efficient way for small and medium-sized software development enterprises to evaluate and improve their software development methods from the security perspective. Based on our approach, enterprises can situationally adapt their software development methods to fit their needs regarding security, agility, and costs. If the primary focus is on the cost efficiency of software development or retaining its agility, most artifacts and several activities may be suitable while roles should generally be avoided. If security, however, is an absolute priority, roles provide most security even though they may not be too cost-efficient. The readers should note that security elements may be related to each other. For example, a security guru can provide security reports, security policies, both, or neither of them. By summarizing the impacts of individual security elements, we may obtain a good estimation of the impact of implementing several security elements. Nevertheless, such an estimation is not perfect since the overall impacts are rarely the sum of individual impacts.

Second, we assessed provided security, cost efficiency, and retained agility of individual security element groups in an international study among practitioners. Practitioners can thus take advantage of this evaluation to make informed decisions regarding which elements to include in their software development methods and projects depending on their specific needs. The placement of security elements into quadrants further facilitates such decision making. For example, practitioners may consider security elements in the *Possible overkill* quadrant if security is a top priority and high costs or compromised agility

can be tolerated. However, security elements in the *Goldilocks* and *Prospects* quadrants may be best suitable for most projects, and especially for small and medium-sized enterprises.

6.3. Limitations

This study has some limitations that the readers should note. First, the literature review was not conducted systematically. Although we were searching the most impactful bibliographic databases and employed the snowball method to identify the studies, we might have missed some existing agile software development methods, especially if not present in the literature. Second, the study focused on agile secure software development. Future work may investigate security elements beyond those already known to be applicable to agile methods. For example, future studies may investigate elements from traditional secure software development to determine if and how they perform when used in agile methods. Such studies may be especially beneficial in identifying prospective security elements besides existing ones. Third, the results indicate a relatively low diversity among different security element groups according to cost efficiency and particularly retained agility. Such distribution of data points in scatter plots can be associated with the fact that approximately 65 percent of our respondents work in the same job position, i.e., as software developers. A more evenly distributed sample of software developers, project managers, security experts, and other software consultants may produce higher response diversity. Fourth, some roles, artifacts, and activities may be related, and this may have affected the scores given by practitioners in our survey. However, such relations cannot be claimed for most security elements, as can be observed by the dispersion of security elements in the scatter plots. Fifth, this paper investigated security elements for a generic agile method under the assumption that the used agile method does not affect the outcomes of implementing security elements (i.e., security, retained agility, cost efficiency). Since the outcomes of implementing security elements in the context of specific agile methods (e.g., Scrum, XP) may nevertheless differ, at least for some security elements, future studies would be needed to investigate whether this is the case and for which security elements.

7. Conclusions and Future Work

This paper has identified and categorized security elements proposed in the literature on agile secure software development. The security elements were then assessed for similarity and grouped in 16 distinct security element groups. Security element groups were then evaluated by practitioners from the software development industry for their cost efficiency, compromised agility, and added security. The results indicate that roles might provide the most security but have the highest costs and most compromised agility. On the other hand, artifacts could contribute to the overall security of the end product the least. However, they are by far the most cost-efficient and retain the most agility. Activities lie in between roles and artifacts in all three key dimensions.

We then used these results to develop a lightweight approach for the evaluation of secure software development method elements. The approach is ready-made for implementation and does not require specialized knowledge or rules to follow. The approach can help software development practitioners to adapt their development methods by prioritizing the adoption of new security elements in their development practice according to their needs and available resources.

There are several directions that future work may undertake. For example, future works could investigate additional situational factors related to specific industry contexts. Unique characteristics of some industries, such as automotive, aviation, and aerospace industries, may require a very strong security undertaking, not characteristic of the majority of other industries. Evaluation of security elements individually allows enterprises to adapt their existing software development methods incrementally, by including a single most suitable element at a time according to their needs. Hence, the results of our study may act as a priority checklist for enterprises attempting to improve their software development methods in terms of software security. Future works may further aim to compare and

evaluate the use of complete off-the-shelf secure software development methods. Such a holistic approach in which all software development lifecycle stages and architectural levels are considered may provide an additional view on the usability and appropriateness of particular methods when compared to each other. To objectively measure the level of software security and enhance the formal software development in practice, specialized security metrics should be additionally investigated, developed, and proposed in the future.

This study took a quantitative approach to evaluate the impact of implementing security elements. Future studies may employ qualitative research designs, such as interviews with professionals, focus groups, or the Delphi method, to gain further insights into the positioning of security element groups into specific quadrants. For example, it may be possible to determine whether certain security elements are firmly positioned in their quadrants or not (e.g., due to some contextual factors).

Author Contributions: Conceptualization, A.M., T.H. and S.V.; methodology, A.M., S.V. and T.H.; validation, T.H. and S.V.; formal analysis, A.M.; investigation, A.M., S.V. and T.H.; resources, S.V.; data curation, A.M.; writing—original draft preparation, A.M.; writing—review and editing, S.V. and T.H.; visualization, A.M.; supervision, S.V. and T.H.; funding acquisition, S.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Faculty of criminal justice and security, University of Maribor.

Institutional Review Board Statement: The study was approved by the Ethics Committee of Faculty of criminal justice and security, University of Maribor, on 8 April 2022.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Dataset underlying the research can be found at Mendeley Data repository. Mihelič, Anže; Vrhovec, Simon; Hovelja, Tomaž (2022), "Evaluation of security elements for agile secure software engineering", Mendeley Data, V1, doi: 10.17632/pbywpngwjw.1 (accessed on 10 November 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bianchi, M.J.; Conforto, E.C.; Amaral, D.C. Beyond the agile methods: A diagnostic tool to support the development of hybrid models. *Int. J. Manag. Proj. Bus.* **2021**, *ahead-of-print*. [CrossRef]
2. Saeedi, K.; Visvizi, A. Software development methodologies, heis, and the digital economy. *Educ. Sci.* **2021**, *11*, 73. [CrossRef]
3. Mihelič, A.; Hovelja, T.; Vrhovec, S.L.R. Towards a delegation-type secure software development method. In Proceedings of the Third Central European Cybersecurity Conference, Munich, Germany, 14–15 November 2019. [CrossRef]
4. Nowroozi, A.; Teymoori, P.; Ramezanifarkhani, T.; Besharati, M.R.; Izadi, M. A Crisis Situations Decision-Making Systems Software Development Process with Rescue Experiences. *IEEE Access* **2020**, *8*, 59599–59617. [CrossRef]
5. Oueslati, H.; Rahman, M.M.; Othmane, L. ben Literature Review of the Challenges of Developing Secure Software Using the Agile Approach. In Proceedings of the 10th International Conference on Availability, Reliability and Security, Toulouse, France, 24–27 August 2015; pp. 540–547.
6. Rindell, K.; Ruohonen, J.; Holvitie, J.; Hyrynsalmi, S.; Leppänen, V. Security in agile software development: A practitioner survey. *Inf. Softw. Technol.* **2021**, *131*, 106488. [CrossRef]
7. Adelyar, S.H.; Norta, A. Towards a Secure Agile Software Development Process. In Proceedings of the 10th International Conference on the Quality of Information and Communications Technology (QUATIC), Lisbon, Portugal, 6–9 September 2016; pp. 101–106.
8. Pohl, C.; Hof, H.-J. Secure Scrum: Development of Secure Software with Scrum. In Proceedings of the The Ninth International Conference on Emerging Security Information, Systems and Technologies Secure, Venice, Italy, 23–28 August 2015; pp. 15–20.
9. Cico, O.; Jaccheri, L.; Nguyen-Duc, A.; Zhang, H. Exploring the intersection between software industry and Software Engineering education—A systematic mapping of Software Engineering Trends. *J. Syst. Softw.* **2021**, *172*, 110736. [CrossRef]
10. ISO/IEC 15408-1:2009. International Organization for Standardization. 2009. Available online: https://www.iso.org/obp/ui/#iso:std:50341:en (accessed on 3 August 2022).
11. Poth, A.; Kottke, M.; Middelhauve, K.; Mahr, T.; Riel, A. Lean integration of it security and data privacy governance aspects into product development in agile organizations. *J. Univers. Comput. Sci.* **2021**, *27*, 868–893. [CrossRef]
12. Soualmi, A.; Laouamer, L.; Alti, A. Performing Security on Digital Images. In *Exploring Security in Software Architecture and Design*; IGI Global: Hershey, PA, USA, 2019; pp. 211–246, ISBN 9781522563136.

13. Tøndel, I.A.; Jaatun, M.G. Towards a Conceptual Framework for Security Requirements Work in Agile Software Development. *Int. J. Syst. Softw. Secur. Prot.* **2020**, *11*, 33–62. [CrossRef]
14. Türpe, S.; Poller, A. Managing security work in scrum: Tensions and challenges. In Proceedings of the CEUR Workshop Proceedings, Bloomington, IN, USA, 20–21 January 2017; pp. 34–49.
15. Ansari, M.T.J.; Al-Zahrani, F.A.; Pandey, D.; Agrawal, A. A fuzzy TOPSIS based analysis toward selection of effective security requirements engineering approach for trustworthy healthcare software development. *BMC Med. Inform. Decis. Mak.* **2020**, *20*, 1–14. [CrossRef]
16. Nina, H.; Pow-Sang, J.A.; Villavicencio, M. Systematic mapping of the literature on Secure Software Development. *IEEE Access* **2021**, *9*, 36852–36867. [CrossRef]
17. Bishop, D.; Rowland, P. Agile and Secure Software Development: An Unfinished Story. *Issues Inf. Syst.* **2019**, *20*, 144–156.
18. Aljaz, T. Improving throughput and due date performance of IT DevOps teams. *Elektrotehniski Vestn. Electrotech. Rev.* **2021**, *88*, 121–128.
19. Hering, D.; Schwartz, T.; Boden, A.; Wulf, V. Integrating usability-engineering into the software developing processes of SME: A case study of software developing SME in Germany. In Proceedings of the 8th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2015, Florence, Italy, 18 May 2015; pp. 121–122.
20. Boden, A.; Nett, B.; Wulf, V. Operational and Strategic Learning in Global Software Development. *IEEE Softw.* **2010**, *27*, 58–65. [CrossRef]
21. Uludag, O.; Putta, A.; Paasivara, M.; Matthes, F. Evolution of the Agile Scaling Frameworks. In Proceedings of the 22nd International Conference on Agile Software Development: Agile Proceses in Software Engineering and Extreme Programming, Virtual Event, 14–18 June 2021; pp. 123–139, ISBN 9781118104354.
22. Pan Fagerlin, W.; Lövstål, E. Top managers' formal and informal control practices in product innovation processes. *Qual. Res. Account. Manag.* **2020**, *17*, 497–524. [CrossRef]
23. Song, M.; Wang, P.; Yang, P. Promotion of secure software development assimilation: Stimulating individual motivation. *Chin. Manag. Stud.* **2018**, *12*, 164–183. [CrossRef]
24. Knight, J.C. Safety critical systems: Challenges and directions. In Proceedings of the 24th International Conference on Software Engineering. ICSE 2002, Orlando, FL, USA, 19–25 May 2002; pp. 547–550.
25. Kasauli, R.; Knauss, E.; Kanagwa, B.; Nilsson, A.; Calikli, G. Safety-critical systems and agile development: A mapping study. In Proceedings of the 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018, Prague, Czech Republic, 29–31 August 2018; pp. 470–477.
26. Inayat, I.; Salim, S.S.; Marczak, S.; Daneva, M.; Shamshirband, S. A systematic literature review on agile requirements engineering practices and challenges. *Comput. Hum. Behav.* **2015**, *51*, 915–929. [CrossRef]
27. Medeiros, J.D.R.V.; Alves, D.C.P.; Vasconcelos, A.; Silva, C.; Wanderley, E. Requirements engineering in agile projects: A systematic mapping based in evidences of industry. In Proceedings of the CIBSE 2015—XVIII Ibero-American Conference on Software Engineering, Lima, Peru, 22–24 April 2015; pp. 460–473.
28. Heikkila, V.T.; Damian, D.; Lassenius, C.; Paasivaara, M. A Mapping Study on Requirements Engineering in Agile Software Development. In Proceedings of the 41st Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2015, Madeira, Portugal, 26–28 August 2015; pp. 199–207.
29. Curcio, K.; Navarro, T.; Malucelli, A.; Reinehr, S. Requirements engineering: A systematic mapping study in agile software development. *J. Syst. Softw.* **2018**, *139*, 32–50. [CrossRef]
30. Mellado, D.; Blanco, C.; Sánchez, L.E.; Fernández-Medina, E. A systematic review of security requirements engineering. *Comput. Stand. Interfaces* **2010**, *32*, 153–165. [CrossRef]
31. Khan, N.F.; Ikram, N. Security requirements engineering: A systematic mapping (2010–2015). In Proceedings of the 2016 International Conference on Software Security and Assurance, ICSSA 2016, Pölten, Austria, 24–25 August 2017; pp. 31–36.
32. Mourao, E.; Kalinowski, M.; Murta, L.; Mendes, E.; Wohlin, C. Investigating the Use of a Hybrid Search Strategy for Systematic Reviews. In Proceedings of the International Symposium on Empirical Software Engineering and Measurement, Toronto, ON, Canada, 9–10 November 2017; pp. 193–198.
33. Mihelič, A.; Vrhovec, S.; Hovelja, T. Sistematični pregled literature agilnih in vitkih pristopov k razvoju varne programske opreme. *Uporab. Inform.* **2020**, *28*, 161–169. [CrossRef]
34. Rindell, K.; Hyrynsalmi, S.; Leppänen, V. Busting a myth: Review of agile security engineering methods. In Proceedings of the ACM International Conference Proceeding Series, Hong Kong, China, 28–30 December 2017; pp. 1–10.
35. Villamizar, H.; Kalinowski, M.; Viana, M.; Fernández, D.M. A systematic mapping study on security in agile requirements engineering. In Proceedings of the 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018, Prague, Czech Republic, 29–31 August 2018; pp. 454–461.
36. Barbosa, D.A.; Sampaio, S. Guide to the Support for the Enhancement of Security Measures in Agile Projects. In Proceedings of the 6th Brazilian Workshop on Agile Methods, WBMA 2015, Pernambuco, Brazil, 21–23 October 2015; pp. 25–31.
37. Myrbakken, H.; Colomo-Palacios, R. DevSecOps: A Multivocal Literature Review. In *Communications in Computer and Information Science*; Mas, A., Mesquida, A., O'Connor, R., Rout, T., Dorling, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2017; Volume 770, pp. 30–42, ISBN 978-3-319-67383-7.

38. Riisom, K.R.; Hubel, M.S.; Alradhi, H.M.; Nielsen, N.B.; Kuusinen, K.; Jabangwe, R. Software security in agile software development: A literature review of challenges and solutions. In Proceedings of the ACM International Conference Proceeding Series, Tokyo, Japan, 25–28 November 2018; pp. 1–5.
39. Migues, S.; Erlikhman, E.; Ewers, J.; Nassery, K. Building Security in Maturity Model (BSIMM) Foundations Report—Version 12; 2021. Available online: https://www.bsimm.com/download.html (accessed on 12 February 2022).
40. Grenning, J. Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods Renaiss. Softw. Consult.* **2002**, *3*, 22–23.
41. Platonova, V.; Bērziša, S. Gamification in Software Development Projects. *Inf. Technol. Manag. Sci.* **2017**, *20*, 58–63. [CrossRef]
42. Williams, L.; Meneely, A. Protection poker: The New Software Security "Game". *IEEE Secur. Priv.* **2010**, *8*, 14–20. [CrossRef]
43. Rygge, H.; Jøsang, A. Threat Poker : Solving Security and Privacy Threats in Agile Software Development. In Proceedings of the 23rd Nordic Conference on Secure IT Systems, Oslo, Norway, 28–30 November 2018; pp. 1–15.
44. Rindell, K.; Hyrynsalmi, S.; Leppänen, V. Securing scrum for VAHTI. In Proceedings of the CEUR Workshop Proceedings, Maribor, Slovenia, 8–10 June 2015; pp. 236–250.
45. Othmane, L.; Angin, L.; Weffers, H.; Bhargava, B. Extending the Agile Development Process to Develop Acceptably Secure Software. *IEEE Trans. Dependable Secur. Comput.* **2014**, *11*, 497–509. [CrossRef]
46. Giacalone, M.; Paci, F.; Mammoliti, R.; Perugino, R.; Massacci, F.; Selli, C. Security Triage: An Industrial Case Study on the Effectiveness of a Lean Methodology to Identify Security Requirements. In Proceedings of the Symposium on Empirical Software Engineering and Measurement—ESEM 2014, Torino, Italy, 18–19 September 2014; pp. 1–8.
47. Maria, R.E.; Rodrigues, L.A.; Pinto, N.A. ScrumS—A model for safe agile development. In Proceedings of the 7th International ACM Conference on Management of Computational and CollEctive Intelligence in Digital EcoSystems, MEDES 2015, New York, NY, USA, 25–29 October 2015; pp. 43–47.
48. Tøndel, I.A.; Cruzes, D.S.; Jaatun, M.G.; Rindell, K. The Security Intention Meeting Series as a way to increase visibility of software security decisions in agile development projects. In Proceedings of the International Conference on Availability, Reliability and Security, Canterbury, UK, 26–29 August 2019; ACM Press: Canterbury, UK, 2019; pp. 1–8.
49. Daud, M.I. Secure software development model: A guide for secure software life cycle. In Proceedings of the International MultiConference of Engineers and Computer Scientists 2010, IMECS 2010, Hong Kong, China, 17–19 March 2010; pp. 724–728.
50. Maier, P.; Ma, Z.; Bloem, R. Towards a Secure SCRUM Process for Agile Web Application Development. In Proceedings of the 12th International Conference on Availability, Reliability and Security—ARES '17, Vienna, Austria, 23–26 August 2017; pp. 1–8.
51. Othmane, L.B.; Angin, P.; Bhargava, B. Using assurance cases to develop iteratively security features using scrum. In Proceedings of the 9th International Conference on Availability, Reliability and Security, ARES 2014, Fribourg, Switzerland, 8–12 September 2014; pp. 490–497.
52. Koc, G.; Aydos, M.; Tekerek, M. Evaluation of Trustworthy Scrum Employment for Agile Software Development based on the Views of Software Developers. In Proceedings of the UBMK 2019—Proceedings, 4th International Conference on Computer Science and Engineering, Samsun, Turkey, 11–15 September 2019; pp. 63–67.
53. Firdaus, A.; Ghani, I.; Jeong, S.R. Secure Feature Driven Development (SFDD) Model for Secure Software Development. In Proceedings of the Procedia—Social and Behavioral Sciences, Iasi, Romania, 10–12 April 2014; Elsevier B.V.: Amsterdam, The Netherlands, 2014; Volume 129, pp. 546–553.
54. Baca, D.; Boldt, M.; Carlsson, B.; Jacobsson, A. A Novel Security-Enhanced Agile Software Development Process Applied in an Industrial Setting. In Proceedings of the ARES Conference International Conference on Availability, Reliability and Security 2015, Toulouse, France, 24–27 August 2015; pp. 11–19.
55. Singhal, S.; Singhal, A. Development of Agile Security Framework Using a Hybrid Technique for Requirements Elicitation. In *Advances in Computing, Communication and Control*; Unnikrishnan, S., Surve, S., Bhoir, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 178–188. ISBN 978-3-642-18439-0.
56. Boström, G.; Wäyrynen, J.; Bodén, M.; Beznosov, K.; Kruchten, P. Extending XP practices to support security requirements engineering. In Proceedings of the 2006 international workshop on Software engineering for secure systems—SESS '06, Shanghai, China, 20–21 May 2006; pp. 11–17.
57. Azham, Z.; Ghani, I.; Ithnin, N. Security backlog in scrum security practices. In Proceedings of the 5th Malaysian Conference in Software Engineering, MySEC 2011, Johor Bahru, Malaysia, 13–14 December 2011; pp. 414–417.
58. Hope, P.; McGraw, G.; Anton, A.I. Misuse and abuse cases: Getting past the positive. *IEEE Secur. Priv.* **2004**, *2*, 90–92. [CrossRef]
59. Musa, S.B.; Norwawi, N.M.; Selamat, M.H.; Sharif, K.Y. Improved extreme programming methodology with inbuilt security. Proceedings of 2011 IEEE Symposium on Computers & Informatics, Kuala Lumpur, Malaysia, 20–23 March 2011. [CrossRef]
60. Tomanek, M.; Klima, T. Penetration Testing in Agile Software Development Projects. *Int. J. Cryptogr. Inf. Secur.* **2015**, *5*, 1–7. [CrossRef]
61. Ge, X.; Paige, R.; Polack, F.; Brooke, P. Extreme Programming Security Practices. In Proceedings of the Agile Processes in Software Engineering and Extreme Programming, Como, Italy, 18–22 June 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 226–230.
62. Siiskonen, T.; Sars, C.; Vah Sipila, A.; Pietikain, A. Generic Security User Stories. In *Handbook of the Secure Agile Software Development Life Cycle*; Pietikinen, P., Rning, J., Eds.; University of Oulu: Oulu, Finland, 2014; pp. 9–14.
63. Lee, K.H.; Park, Y.B. Adaption of integrated secure guide for secure software development lifecycle. *Int. J. Secur. Its Appl.* **2016**, *10*, 145–154. [CrossRef]

64. Ionita, D.; Van Der Velden, C.; Ikkink, H.K.; Eelko, N. Towards Risk-Driven Security Requirements Management in Agile Software Development. *Lect. Notes Bus. Inf. Process.* **2019**, *350*, 133–144. [CrossRef]
65. Stålhane, T.; Myklebust, T.; Hanssen, G. The application of safe scrum to IEC 61508 certifiable software. In Proceedings of the 11th International Probabilistic Safety Assessment and Management Conference and the Annual European Safety and Reliability Conference 2012, PSAM11 ESREL 2012, Helsinki, Finland, 25–29 June 2012; Volume 8, pp. 6052–6061.
66. Stålhane, T.; Johnsen, S.O. Resilience and safety in agile development (Through safescrum). In Proceedings of the Safety and Reliability—Theory and Applications—Proceedings of the 27th European Safety and Reliability Conference, ESREL 2017, Portoroz, Slovenia, 18–22 June 2017; pp. 945–954.
67. Nguyen, J.; Dupuis, M. Closing the feedback loop between UX design, software development, security engineering, and operations. In Proceedings of the 20th Annual Conference on Information Technology Education—SIGITE 2019, Tacoma, WA, USA, 3–5 October 2019; pp. 93–98.
68. de Vicente Mohino, J.; Higuera, J.B.; Higuera, J.R.B.; Montalvo, J.A.S. The application of a new secure software development life cycle (S-SDLC) with agile methodologies. *Electronics* **2019**, *8*, 1218. [CrossRef]
69. Mougouei, D.; Sani, N.F.M.; Almasi, M.M. S-Scrum : A Secure Methodology for Agile Development of Web Services. *World Comput. Sci. Inf. Technol. J. (WSCIT)* **2013**, *3*, 15–19.
70. Tappenden, A.F.; Huynh, T.; Miller, J.; Geras, A.; Smith, M. Agile Development of Secure Web-Based Applications. *Int. J. Inf. Technol. Web Eng. (IJITWE)* **2006**, *1*, 1–24. [CrossRef]
71. Yu, W.D.; Le, K. Towards a secure software development lifecycle with SQUARE+R. In Proceedings of the International Computer Software and Applications Conference, Singapore, 9–10 June 2012; pp. 565–570.
72. Ghani, I.; Azham, Z.; Jeong, S.R. Integrating software security into agile-Scrum method. *KSII Trans. Internet Inf. Syst.* **2014**, *8*, 646–663. [CrossRef]
73. Fowler, F.M. *Navigating Hybrid Scrum Environments*; Apress: Sunnyvale, CA, USA, 2019; ISBN 9781484241639.
74. Kline, R.B. *Principles and Practice of Structural Equation Modeling*; Guilford Press: New York, NY, USA, 2011.
75. Karim, N.S.A.; Albuolayan, A.; Saba, T.; Rehman, A. The practice of secure software development in SDLC: An investigation through existing model and a case study. *Secur. Commun. Netw.* **2016**, *9*, 5333–5345. [CrossRef]
76. Ansari, M.T.J.; Pandey, D.; Alenezi, M. STORE: Security Threat Oriented Requirements Engineering Methodology. *J. King Saud Univ. Comput. Inf. Sci.* **2018**, *34*, 191–203. [CrossRef]
77. *Veracode Secure Coding Best Practices Handbook*; Veracode: Burlington, MA, USA, 2021.
78. Tøndel, I.A.; Jaatun, M.G.; Cruzes, D.S.; Moe, N.B. Risk Centric Activities in Secure Software Development in Public Organisations. *Int. J. Secur. Softw. Eng.* **2018**, *8*, 1–30. [CrossRef]