applied sciences

Article

Towards the Integration of Security Practices in Agile Software Development: A Systematic Mapping Review

Yolanda Valdés-Rodríguez [1,†], Jorge Hochstetter-Diez [2,*,†], Jaime Díaz-Arancibia [2,†] and Rodrigo Cadena-Martínez [3,4,†]

1 Universidad Autónoma de Chile, 5 Poniente, Talca 1670, Chile
2 Universidad de La Frontera, Francisco Salazar, Temuco 01145, Chile
3 Universidad Tecnológica de México, Marina Nacional 180, Anahuác I Sección, Ciudad de México 11320, Mexico
4 Universidad Americana de Europa, Av. Bonampak Sm. 6-Mz. 1, Lt. 1, Cancún, Quintana Roo 77500, Mexico
* Correspondence: jorge.hochstetter@ufrontera.cl; Tel.: +56-45-2-744217
† These authors contributed equally to this work.

Abstract: Software development must be based on more than just the experience and capabilities of your programmers and your team. The importance of obtaining a quality product lies in the risks that can be exploited by software vulnerabilities, which can jeopardize organizational assets, consumer confidence, operations, and a broad spectrum of applications. Several methods, techniques, and models have been suggested and developed to address software security. However, only a few have solid evidence for creating secure software applications. The main objective of this paper is to survey the literature for methods or models suitable for considering the integration of security in all or some of the phases of the software development life cycle and which ones are most considered or neglected. This study represents the beginning of research to generate a methodology that integrates security practices in agile software development, allowing inexperienced developers to create more secure applications.

Keywords: secure development; secure software; software process; agile methodology

check for updates

Citation: Valdés-Rodríguez, Y.; Hochstetter-Diez, J.; Díaz-Arancibia, J.; Cadena-Martínez, R. Towards the Integration of Security Practices in Agile Software Development: A Systematic Mapping Review. *Appl. Sci.* **2023**, *13*, 4578. https://doi.org/10.3390/app13074578

Academic Editor: Gianluca Lax

Received: 27 February 2023
Revised: 20 March 2023
Accepted: 28 March 2023
Published: 4 April 2023

1. Introduction

Advances in information and communication technologies allow people and organizations to have greater connectivity [1]. Bringing this access to the masses has specific risks associated with its use, especially considering that we rely heavily on software systems in various daily activities we may perform [1,2]. Due to that, it is relevant to ensure various security issues [3]; for example, we have to expect that the software will continue to function correctly under malicious attack [4].

Secure software is designed, implemented, configured, and operated to fulfill essential properties: to continue functioning in the presence of computer attacks or mitigate damage and recover as quickly as possible [5]. Software developers must design, develop and deploy our systems with a secure mindset, applying strategies that minimize the likelihood of exposure and impact to threats [6,7]. However, in practice, this is different; software development is based on adopting a reactive approach, which consists of assessing the security of applications once they have been developed, focusing on the later stages of the software life cycle. Fixing bugs in this way helps; however, this approach proves to be more costly when resolving software security flaws at later stages [8].

Thus, organizations have been forced to define and implement a set of countermeasures that allow us to secure our information assets against casual or deliberate attacks. Yet, these seem insufficient in the face of the increase in this type of attack worldwide, making it essential for organizations to define security policies that consider the wide variety of attacks to which they are exposed [9,10].

The traditional response to this scenario focuses on reducing risk by standardizing information security and defining and implementing best practices or controls.

For example, the ISO27001 [11], NIST [12], and COBIT [13] standards propose a set of controls that must be complied with to secure an organization's information assets and operational continuity.

Although it has been demonstrated that the implementation of the security controls of these standards reduces security incidents, what is really implemented is a layer or security shield on top of the existing systems, which means the software systems themselves are not aware of the anomalous behaviors of the users, and, therefore, as software systems, they do not have a reaction protocol for it, that is, once the standards penetrate the security shield provided (if it exists), the software systems are exposed [10,14].

A complementary approach is the determination of the particular security requirements of each organization, reflected as non-functional requirements of their software systems [15,16]. From this point, the different phases of the software production cycle must accommodate these requirements, which means that security aspects must be represented as requirements; they must be considered in the design, in the development of test cases, in the coding, and application of tests, packaging, and delivery of the product [10,17].

A software product can be vulnerable to its construction failures or provoked attacks [18,19]. To reduce vulnerabilities and make the product secure, measures such as the integration of security concepts in all its development stages or approaches or methodologies that allow the integration of security into the software life-cycle must be applied [18,20].

This article aims to conduct a general literature review to identify approaches or frameworks applied to secure software development. With this work, we contribute to understanding which phases of the software development cycle are the least addressed and how these proposals have been developed. To do so, we apply the systematic mapping methodology that provides an overview of a research area through classification. We share the current trends of existing frameworks for software development involving security and provide a scenario and resources to researchers that enable the creation of a general framework proposal that covers the main security challenges in the software life cycle and can be used by developers without security expertise.

Unfortunately, engineering programs still do not address the importance of including security in software design and development [21]. Most engineering design methodologies do not consider software-related risks, considering that security tasks are the responsibility of the IT area and keeping the developed engineering systems secure [21].

From a software engineering perspective, our contribution to the theory is the basis for incorporating security practices into software development that inexperienced developers can use on an individual basis.

This paper is structured as follows: Section 2 presents the background, Section 3 Security Practices in Agile Software Development, Section 4 related work, Section 5 presents the working methodology, Section 6 the main results together with the research gaps and answers to the questions posed, Section 7 our proposal and Section 8 the limitations of the study. Finally, conclusions and future work are presented.

2. Background

According to [22], the software development life cycle (SDLC) is defined as a set of phases that software must go through from the moment a need arises, passing through the phases of development and operation.

Software engineering implements a series of "models" that divide the project into stages from its initial conception, development, testing, release, and maintenance. For each stage, standards are created those guide engineers, guiding the work of the different phases in technical activities and, thus, providing a formal framework for management, development progress, maintenance, and resource estimation.

Developing secure and reliable software also requires adopting a systematic process or discipline that addresses security in each of the phases of the software life cycle.

Some software development models integrate security activities such as specific design principles and security practices.

- Secure Software Development Life Cycle (S-SDLC), this model is based on verifying security requirements throughout the different phases of software construction [23,24]. The advantage of adopting an S-SDLC approach is the identification of coding and design errors in the early stages of development [24].
  The research paper by Mohino [23] proposes a new Secure Software Development Life Cycle (S-SDLC) that addresses security issues in software development. The S-SDLC includes six phases: requirements, design, implementation, testing, deployment, and maintenance, and integrates specific security activities into each phase. The author emphasizes the importance of incorporating security early in the SDLC process, beginning with the requirements phase. Developing security requirements is derived from business requirements and security policies to achieve this. During the design phase, the focus is on creating a secure software architecture that aligns with the security requirements. The implementation phase includes secure coding practices that aim to prevent security vulnerabilities. Security testing is carried out during the testing phase to identify and remedy security issues. The deployment phase involves secure deployment and configuration of the software to ensure the software remains secure. The maintenance phase focuses on ensuring the software continues to be secure through ongoing monitoring, vulnerability management, and incident response. The proposed S-SDLC provides a comprehensive approach to integrating security into the SDLC process to develop secure software from the outset. With S-SDLC, several additional security practices and activities can be enriched. These practices include a security specification language, a security requirements engineering process, a secure design specification language, a set of secure design guidelines, a secure design pattern, a secure coding standard, and a software security assurance method, which may comprise penetration testing, static analysis for security, and code reviews for security [23].
- Security Assurance Maturity Model (SAMM), the goal of the OWASP SAMM is to be the leading maturity model for software assurance that provides a practical and measurable way for all types of organizations to analyze and improve their software security posture [25]. This model supports the entire software lifecycle, including development and acquisition, and is independent of technology and processes. The Software Security Maturity Model (SAMM) is an open methodology that allows organizations to design and implement a strategy to improve software security [26]. This model addresses the specific software security risks faced by each organization. The Security Assurance Maturity Model (SAMM) is a framework designed to help organizations to improve their software security processes. SAMM has four domains: Governance, Construction, Verification, and Deployment. Each domain has three maturity levels, ranging from ad-hoc to optimized processes. The Governance domain covers policy and strategy, which provides direction and guidance for the security program. The Construction domain includes software design, development, and testing. The Verification domain covers testing and analysis to ensure the software is secure. The Deployment domain includes release management, operations, and incident management. SAMM also includes a measurement model, which assesses an organization's maturity level for each domain. The model evaluates the organization's practices, policies, and procedures against the SAMM framework. The results provide a roadmap for improvement. SAMM is a flexible and adaptable framework that allows organizations to tailor the model to their needs. It provides guidance and best practices to improve software security and helps organizations to mature their software security processes over time [26].
- McGraw's Secure Software Development Life Cycle Process, in his article "Security Software Building Security in Seven Touchpoints for Software Security," McGraw's proposal focuses on integrating security into the software development life cycle

(SDLC) through seven touchpoints [27]. The seven touchpoints include (i) Requirements: Define the software security requirements and establish the basis for the rest of the development process. (ii) Design: Design a secure software architecture considering the previously established security requirements. (iii) Implementation: Write lines of code and apply secure coding practices. (iv) Testing: Perform security tests to ensure compliance with previously established security requirements. (v) Integration: Ensure proper software integration with other systems and maintain security. (vi) Deployment: Implement the software securely and configure it properly. (vii) Maintenance: Implement safe maintenance practices to ensure the continued software security and quickly address security issues. This approach focuses on integrating security into all stages of the software development lifecycle to ensure the resulting software is secure and reliable [27].

We can identify several differences between the Secure Software Development Life Cycle (S-SDLC), Security Assurance Maturity Model (SAMM), and McGraw's Secure Software Development Life Cycle Process: (i) Approach: The S-SDLC and McGraw's SDLC process focus on integrating security into the software development life cycle. In contrast, SAMM focuses on improving the organization's overall security program. (ii) Framework Structure: The S-SDLC and McGraw's SDLC process provide a structured approach to software development, while SAMM provides a framework for assessing an organization's overall security maturity. (iii) Domains and Maturity Levels: The S-SDLC has six phases, McGraw's SDLC process has seven touchpoints, and SAMM has four domains, each with three maturity levels. (iv) Emphasis: The S-SDLC and McGraw's SDLC process emphasize the software development process, while SAMM emphasizes the security program. (v) Assessment: The S-SDLC and McGraw's SDLC process do not include an assessment model, while SAMM includes a measurement model that assesses an organization's security maturity level.

3. Security Practices in Agile Software Development

Agile software development is gaining acceptance as a flexible approach. This approach prioritizes the software's continuous and early delivery, changing the requirements even in advanced stages of development and adapting to the customer's needs [28]. Although the agile approach is gaining more and more followers, it reveals that it has certain security-related disadvantages.

Generally, security is not considered in any of the phases of the software development life cycle (SDLC) [29]. At best, it is covered through the definition of non-functional requirements; for this reason, it is either not taken into account, or it is done at the end of the project [23]. Security is seen as an element that increases project development and delivery times, which goes against agile principles [30].

There are some software development lifecycles that are starting to consider agile principles:

- Correctness by Construction (CbyC), is a highly effective method for developing software that requires critical levels of safety and provability. The main objectives of this methodology are to minimize the defect rate and increase resilience to change, achieved through two fundamental principles: making it very difficult to introduce bugs and ensuring that bugs are identified and eliminated as early as possible. To achieve these goals, CbyC seeks to ensure that software is correct from the start through rigorous safety requirements, a detailed definition of system behavior, and a robust and verifiable design [31,32].
- ViewNext, model proposed by [33] is an agile adaptation of the S-SDLC [24], which incorporates security best practices from known models along with other security tasks, based on the spiral model, is integrated into normal software engineering life cycles. The model corrects weaknesses present in previous models and follows a preventive approach, making it an effective alternative for secure software development. Known

as Agile and Secure Software Development Life, this model has been the subject of study in [34].

- Microsoft SDL Agile, it is an adaptation of the SDL Methodology (Security Development Lifecycle) that was developed by Microsoft to integrate security into agile software development processes [35]. The Agile SDL methodology focuses on integrating security into each iteration of the agile software development process. Rather than following a "wait until the end" approach to integrating security, the Agile SDL methodology promotes the inclusion of security activities in all phases of the agile development process. Security activities include early risk identification, defining secure user stories, performing security testing in each iteration, and implementing security best practices in the agile development process. The Agile SDL methodology is based on the agile software development lifecycle, which includes planning, analysis, design, implementation, testing, and maintenance. By integrating security into each stage of this lifecycle, it is possible to ensure that the software developed is secure and complies with security requirements.
- Building Security In Maturity Model (BSIMM), is a security maturity model used to describe the practices and processes used by leading software security organizations to develop, improve and maintain effective software security programs [36]. BSIMM focuses on assessing organizations' software security programs by measuring their maturity in 12 common security practices. This model helps organizations develop their own software security program and provides a tool for ongoing assessment of software security maturity over time. The latest version of the model, BSIMM10, released in 2020, addresses agile properties of software development. It includes practices and processes relevant to agile approaches, such as continuous integration and continuous delivery, security automation, security management in the product backlog, and security collaboration between development teams. In addition, BSIMM10 focuses on the importance of security in the context of agile frameworks, such as Scrum and DevOps.

The software development models presented in Section 2 share a common focus on improving software safety throughout the software development life cycle. BSIMM and SAMM are software security maturity models that measure the maturity of software security programs and provide guidance for improving them, although BSIMM focus specifically on agile properties. Meanwhile, S-SDLC and McGraw's Secure Software Development Life Cycle Process are secure software development life cycle models that integrate security into each stage of the development process. While S-SDLC provides general guidelines and best practices for developing secure software, McGraw's Secure Software Development Life Cycle Process focuses on eliminating vulnerabilities through a secure architecture from the outset. On the other hand, Correctness by Construction is a methodology that seeks to produce correct software from the beginning through a rigorous definition of security requirements, a solid and verifiable design, and a preventive approach to avoid introducing errors. Finally, SDL Agile is an adaptation of Microsoft's SDL model that integrates security into agile software development processes. This model focuses on security automation, security management in the product backlog, and security collaboration between development teams, fostering collaboration and continuous integration of security throughout the software development lifecycle. In summary, while these models share a common goal of improving software security, each offers a unique and complementary approach to achieving it.

Software assurance is the confidence that a system meets all its security requirements. In most of those requirements of interest to customers and users of the software, this confidence is based on specific evidence collected and evaluated through assurance techniques [37].

The techniques or mechanisms established for information security are considered rigid to respond to the changes and advances that are presented in the changing security environment, where there is a need for a more agile method to deal with new threats and vulnerabilities [38]. Thus, the traditionally established security mechanisms are no longer effective when used with software development methodologies adapted to the needs of the current environment, such as agile methodologies [39].

The use of agile methodologies in software development implies, on several occasions, not considering the good practices of secure development, whose purpose is to guarantee the fulfillment of the own security policies of the software development [40–45].

Several authors state that developing secure software using agile methodologies is challenging. Applying security practices in agile methodologies presents challenges because agile methodologies support requirements changes prefer frequent deliveries, and their practices do not include security engineering activities [41].

The paper published by [46] discusses defects in the requirements specification stage, which generally in security aspects are misunderstood and incorrectly specified due to lack of security expertise. These concerns become even more challenging in agile contexts, where lightweight documentation is generally produced. To address this problem, the indicated article proposes an approach to review security-related aspects of web application requirements specifications in agile contexts. The methodology considers user stories as inputs and relates them to the OWASP (Open Web Application Security Project) security properties, which must be verified and then generate a reading technique to help reviewers detect defects. The methodology was evaluated through three experimental tests performed with 56 novice software engineers, measuring effectiveness, efficiency, usability, and ease of use. The results indicate that the proposed methodology has a positive impact on the number of vulnerability findings in terms of effectiveness and efficiency.

There seems to be a clear need for a software development model which addresses security issues at any stage of the software life cycle and considers the benefits of agile models. In this context [23] proposes a model that introduces security as a crucial element in software development environments and, at the same time, leverages agile properties.

On the other hand, Sharma et al. [47] offers a framework for agile development that addresses security, considering customer requirements. The implementation of this Framework has been implemented in Java to automate the whole process, although the author points out that the suggested security activities should be tested and evaluated in a real industrial environment.

4. Related Work

There is an interesting study [48] where the authors address the main reasons why current software systems are so insecure, pointing out the lack of empirical research in the area of Software Engineering to understand better where and why security-critical bugs arise in the software development life cycle. The lack of tools at all levels of the software development cycle to automatically detect coding vulnerabilities, and finally, he points out that a significant factor is that the training of today's students will impact tomorrow's engineers, so both educational institutions and engineering teachers must integrate security into their curricula [49,50].

Here is the importance of generating secure software, especially considering that we rely heavily on software systems in various daily activities that we may perform [1,2]. It is a high priority to ensure several points of security [3] so that the software continues to function properly under malicious attack and does not allow unauthorized access to sensitive data [4] and does not allow unauthorized access to sensitive data.

The main problem in mitigating risks and achieving cybercrime reduction is the unavailability of a single framework that can integrate security and design tactics when building the software [51,52].

Considering the above, it is necessary to know current studies that show trends, empirical studies, or evaluations of security methodologies, tools, or techniques in developing secure software. Under this logic, we have found studies such as Abeyrathna et al. [53], who propose to model the knowledge and find links between security flaws in source code and security flaws in the design phase. The results indicate that security problems at the software architecture level lead to security flaws in the source code.

Regarding the adoption of static analysis for software security evaluation, Nguyen et al. [54] presents the results of a case study in which static application security testing (SAST) was applied in an open-source e-government project. The results show that it is possible to increase performance by combining different SAST tools.

A study on developer discussions and security challenges related to the most popular programming languages is presented in [55]. More than 20 million issues from 27,312 GitHub repositories were examined for this research. Subsequently, an analysis was performed using quantitative and qualitative methods for 15 of the most popular programming languages, such as Java, C, Python, and PHP.

The main findings of this study were: Security issues of web-oriented languages (e.g., JavaScript, TypeScript, and PHP) receive the highest popularity, and mobile-oriented language users (e.g., Java, C#, Objective-C, and Swift) have the highest level of security expertise. Shell-based and Web-oriented languages experience significantly higher average rates of security discussion. Scientific programming-oriented languages (Julia, MATLAB, and R) exhibit very small numbers of security discussions. C/C++ are the only languages that face memory management challenges.

Antal et al. [56] conducts a study on the security awareness of open-source communities by examining Python and JavaScript projects for vulnerabilities. These results allow us to identify categories of vulnerabilities that are not sufficiently addressed and explore patterns that could help build or create vulnerability prediction models.

The work done by [34] stands out for managing security issues from the initial stages of the development process. The author proposes a secure development model through a case study applied to a software development company. The results indicate that the number of vulnerabilities detected is reduced by 66%.

Correa et al. [57] propose a methodology that evaluates and prevents security vulnerabilities in web applications. The analysis process is performed through black-box and white-box evaluations using dynamic and static analysis tools.

The results presented in [58] contribute to understanding how teams integrate and model threats in the context of agile development and what can be done to facilitate the process. They also present a list of recommendations that can help companies improve their threat modeling strategies.

Engineering programs still need to address the importance of including security in software design. In this sense, the work proposed by [21] proposes a practical approach to integrate security in the development lifecycle of engineering systems and support engineering students in developing secure products and processes. The proposed model stems from an adaptation of SecSDM (Secure Software Development Methodology) [21].

Concerning previous studies, a 2018 study on security in agile requirements engineering [59] provides some insights on handling security requirements, modifying agile methods already in use, introducing new artifacts, or introducing guidelines to handle security. The study also identifies significant limitations of agile approaches considering security: lack of time, developer skills and security awareness, and middleware guidelines for gathering and addressing security requirements.

Weir et al. [60] have studied the effectiveness of a series of interventions by a facilitator and a development team that does not require the involvement of a security expert. The results indicate substantial improvements in the ability of teams to deliver secure software and can even be effective with teams with little or no security expertise—the research project aimed at defining a cost-effective solution to support development teams in creating secure products. The results showed that the intervention improved the process or understanding of security in all but the most security-savvy group. The intervention had the most significant impact when the facilitators of the workshops were managers.

On the other hand, the private sector presents interesting initiatives to mitigate the risks associated with integrating Security Practices in Agile Software Development.

BOOST [61], a company dedicated to software development with agile methodologies, presents a working guide for security risks to be managed in software projects using agile methodologies. The company Veracode [62] provides a cloud-based platform for security testing.

Legitsecurity's website Legitsecurity [63] defines the ten security issues of agile software development that we should be aware. Organizations such as OWASP [64] and SANS [65] are constantly working to identify, analyze and determine ways to mitigate software vulnerabilities, contributing in different ways to the community to solve problems due to software vulnerabilities.

5. Methodology

The methodology consists of applying the systematic mapping technique, which provides an overview of a research area through classification [66]. It is a method commonly used to answer, one or more research questions methodically. Next, we present the steps followed by the protocol for conducting systematic mappings, adapting the stages described in Figure 1.

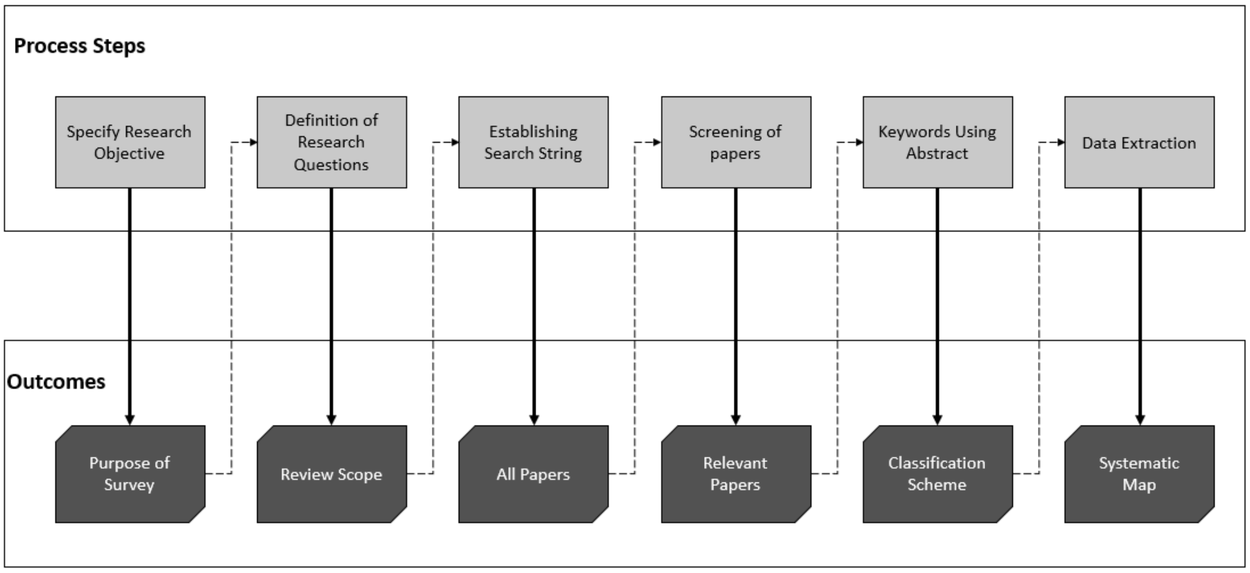

Figure 1. Stages of the systematic mapping process.

The stages that make up the systematic mapping process are described in the following sections.

5.1. Goal and Research Questions

The objective of this systematic mapping is to identify the methods or models that consider the integration of security in all or some of the phases of the software development life cycle.

We define the research questions in such a way as to collect developed works and detect trends associated with the subject of study. The research questions are detailed on Table 1.

5.2. Generating the Search String

A search was carried out using a search string composed of keywords specific to this study to address the RQs and was applied to Data Source, all related to the study topic. The articles were searched using the search string from the PICO (Population, Intervention, Comparison and Results) strategy indicated in [67]. The articles were searched using the search string detailed in Table 2, considering the elements described in the Research questions.

Table 1. Research Questions to be applied.

Research Question	Motivation
RQ1: How many articles are related to secure software development?	Recognize the documents that present proposals related to the initiatives for secure software. This is the first step to be able to answer the following questions.
RQ2: How many articles study security practices for agile software development?	Software development must be based on something more than the experience of its programmers. Recognizing the number of works related to security practices in agile software development helps to categorize the different levels of support from comparison studies to proposals for secure development.
RQ3: What is the context/setting where of the articles take place?	Ascertain the scenarios where the selected investigations are carried out, enables the analysis of the proposals for the use of security controls in software development.
RQ4: What are the SDLC phases that have most addressed security in software development?	Recognize articles that present proposals related to the integration of security controls in one or some of the software construction phases
RQ5: Which particular phase of SDLC has been least discussed and addressed in the literature?	Know which are the phases of software construction that are least addressed in the literature and that can be part of a valuable contribution in subsequent studies
RQ6: What results have the new methods or models yielded to ensure security in developing secure software?	Know the results of previous proposals and determine research gaps for future work in secure development engineering

Table 2. Search string.

Main Concepts	Software security, software secure, software privacy, software safety, software engineering, software development life cycle, SDLC, software security model, software security process, software security methodology
Groups of terms	("Software security" or "software secure" or "software privacy" or "software safety") ("software engineering" or "software development lifecycle" or "SDLC" or "Software security model" or "software security process" or "software security methodology")
Search String	("Software security" or "software secure" or "software privacy" or "software safety") AND ("software engineering" or "software development lifecycle" or "SDLC" or "software security model" or "software security process" or "software security methodology")

5.3. Data Extraction

To carry out the search and data extraction process, the following digital data sources were defined: Web of Science (WoS), ACM Digital Library, Science Direct, IEEE Xplore, and Scopus. These databases allowed searches to download many articles related to the topic. Additionally, search criteria were defined by overview, since all the databases consulted have this option.

5.4. Inclusion and Exclusion Criteria

The selected papers found through the aforementioned data sources were based on the following inclusion/exclusion criteria:

Inclusion criteria: Studies from 2018 onwards; Articles published in conferences and journals; Articles that consider security according to the elements described in the research questions.

Exclusion Criteria: Studies prior to 2018; Literature reviews or states of the art; Duplicate studies in different databases; Articles that address security only as a tool, emphasizing technology, not its use applied to software development; Articles that consider the impact of applying some security metric on software development; Articles that discuss security from the point of view of infrastructure, IoT, physical security, networks.

5.5. Search Execution

The data extraction process was carried out through an abstract search of the selected data sources. Initially, a total of 9810 articles was obtained (see Table 3). After narrowing this search according to the exclusion criteria from 2018, it was reduced to 368 articles. This is because we want to examine new studies. This information was extracted using the export tools of each digital library for further analysis.

Table 3. Articles found.

Data Source	Abstract Selection	Limited to 5 Years
Web of Science	29	20
ACM Digital Library	34	16
Science Direct	0	0
Scopus	241	79
IEEE Xplore	9506	253
Total	9810	368

After extracting the initially selected articles, duplicate studies or studies that were part of a literature review were eliminated, reducing the number further to 312 papers. Then, the inclusion and exclusion criteria were applied by reading the abstracts and finally discarding those articles that: only refer to software development without considering security in any of its phases; address security only as a tool, emphasizing technology, not its use applied to software development; consider the impact of applying some security metric in software development; and increase security from the point of view of infrastructure, IoT, physical security, networks.

5.6. Selection of Articles

After reading the abstract, 51 articles were selected for full reading, of which 12 were not included in the study due to lack of access to them. Finally, the study was conducted with 39 scientific articles that were read in full text to assess their relevance and provide answers to the research questions (see Figure 2).

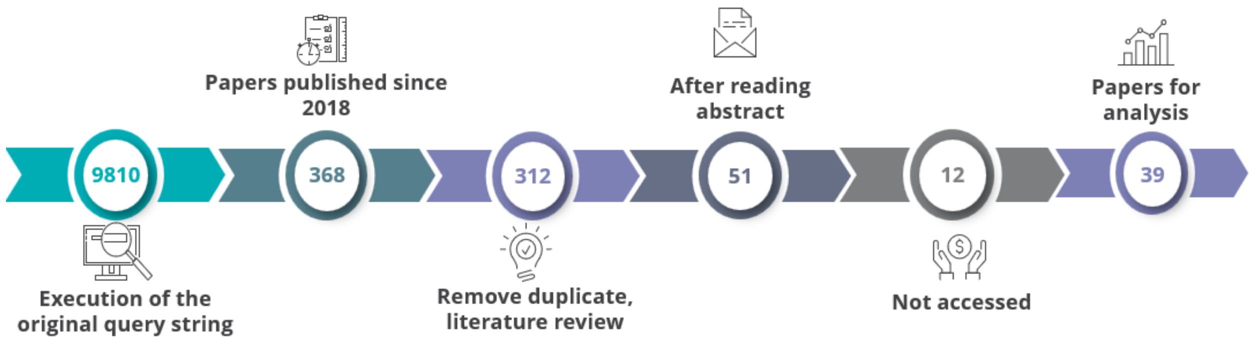
Papers published since 2018
After reading abstract
Papers for analysis
9810
368
312
51
12
39
Execution of the original query string
Remove duplicate, literature review
Not accessed

Figure 2. Search and selection process.

5.7. Classification Scheme

The 39 selected works were classified in three dimensions: by time, by categories and type of work.

The time dimension classifies works according to the year of publication, considering the basis that we have selected works as the last 5 years. The dimension where the works correspond to the context where the research was conducted, with these scenarios being Health, Organizations, Vulnerabilities, Cybersecurity Education, Critical Infrastructure, and Software Industry. It should be noted that the works can be classified into more than one category.

The dimension type of work was classified into 3 types: Implementation: corresponding to works that generate new proposals regarding the development of secure software. Analysis: those works that perform analyses or comparisons of studies related to the development of secure software, or when the works correspond to literature reviews. Use: those works based on applying proposals from other authors on issues related to the development of secure software.

6. Main Results

6.1. Overall Analysis by Characteristics

This section presents the results of the classification and analysis of the selected studies. The first aspect focuses on the type of publication, whether journals or congresses. Most of the study articles were published in different journals (30 articles), which represents 77% of the total articles. The databases of these journals are IEEE Xplore, Web of Science, ACM, and Scopus. On the other hand, the articles published in conferences represent 23% of the total analyzed (9 articles, see Figure 3). This allows us to indicate which type of publication is more attractive to researchers in security aspects.

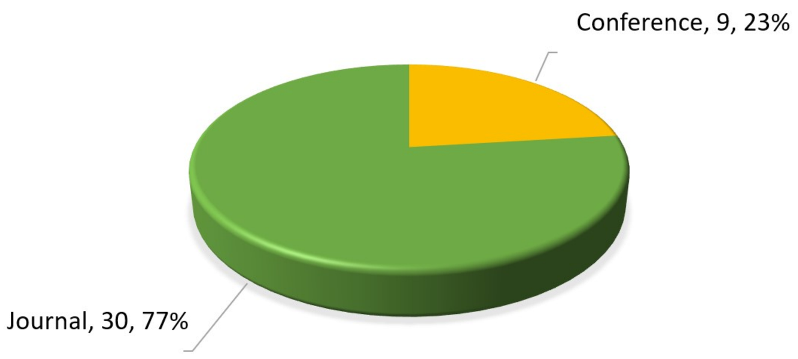

Figure 3. Articles published in journals and conferences.

A second aspect of the classification focuses on the sources of the primary published studies, which helps us to identify which databases are most active in security aspects (see Figure 4). The sources defined for the study correspond to Web of Science, ACM Digital Library, Science Direct, Scopus, and IEEE Xplore; according to the findings, there is a wide preference in the research community for publishing in IEEE Xplore with 51% of the selected works. In second place is Scopus with 26%, Web of Science with 13% of the preferences, and finally ACM Digital Library with 4 selected articles. It is important to mention that in Science Direct, there were no papers contributing to this research, which may be because it is not an interesting and attractive source for researchers in the area.

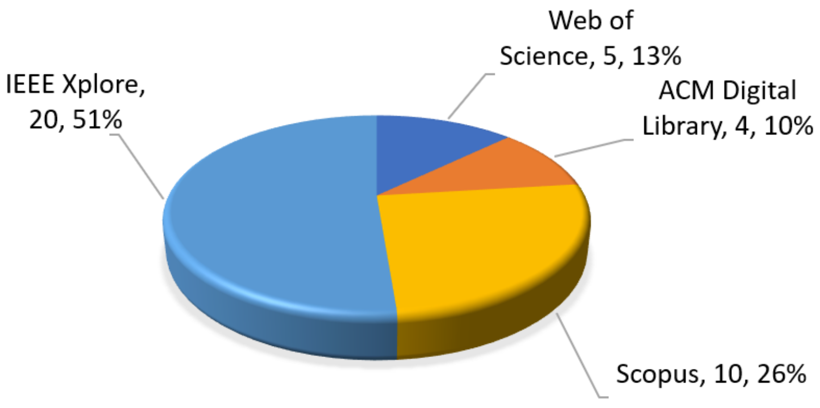

Figure 4. Articles database.

Regarding the years of publication of the selected articles, the following Figure 5 illustrates the distribution of the studies by year of publication. A downward trend is evident since 2020, this is probably due to the pandemic effect. In 2019, there was a considerable increase in publications, with 11 papers. It is important to consider that the search was performed in August 2022, so it is expected that by the end of the year, there may be more publications in the area.

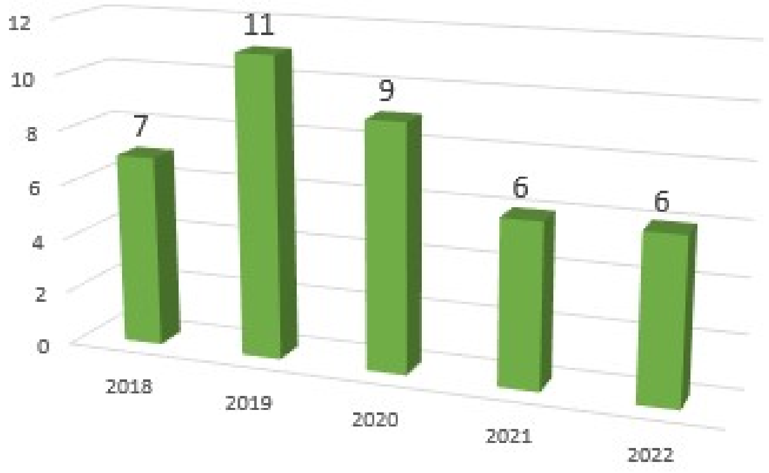

Figure 5. Article publications by years.

Another aspect that needs to be discussed and will provide us with inputs to answer the research questions is to know what is the proportion of articles that contribute to the research topic, specifically on security in the software development cycle.

Of the articles reviewed, it can be seen that just over half of the articles analyzed (54%) use security in the phases of the software development cycle (SDLC), corresponding to 21 articles. However, there is a significant number of articles (26%) that refer to the evaluation of security metrics, comparisons, and risk analysis, among others that do not consider applying security in any phase of the SDLC. Finally, 8 articles were found that do not consider the use of security in the SDLC phases.

The classification of articles was carried out considering:

- Articles that use security in the SDLC phases (see Figure 6),
- Articles that use a methodology or propose a process,
- Articles that evaluate security after applying the proposed method.

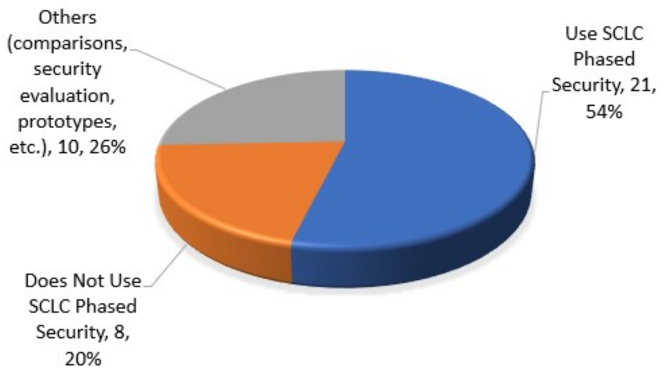

Figure 6. Publications of articles that use security in the phases of SDLC.

The analysis shows that the largest number of papers are concentrated in the implementation category, with 28 (72% of the total). Of these, 17 correspond to the Software Industry category according to the context where they are developed, representing 44% of the total.

Regarding the time dimension of these 17 works, it is evident that their concentration is in 2020 with 6 works, then there is a decline in 2021 and 2022, which may be due to the pandemic. It is important to mention that the data extraction was performed in August 2022, so it can be inferred that at the end of the year, this amount should be higher.

6.2. Systematic Mapping

We have built a map to represent and analyze the selected works. Figure 7 shows those works that we have classified in the three dimensions; on the left, we group by type of work, on the right by year, and in the central column, the categories.

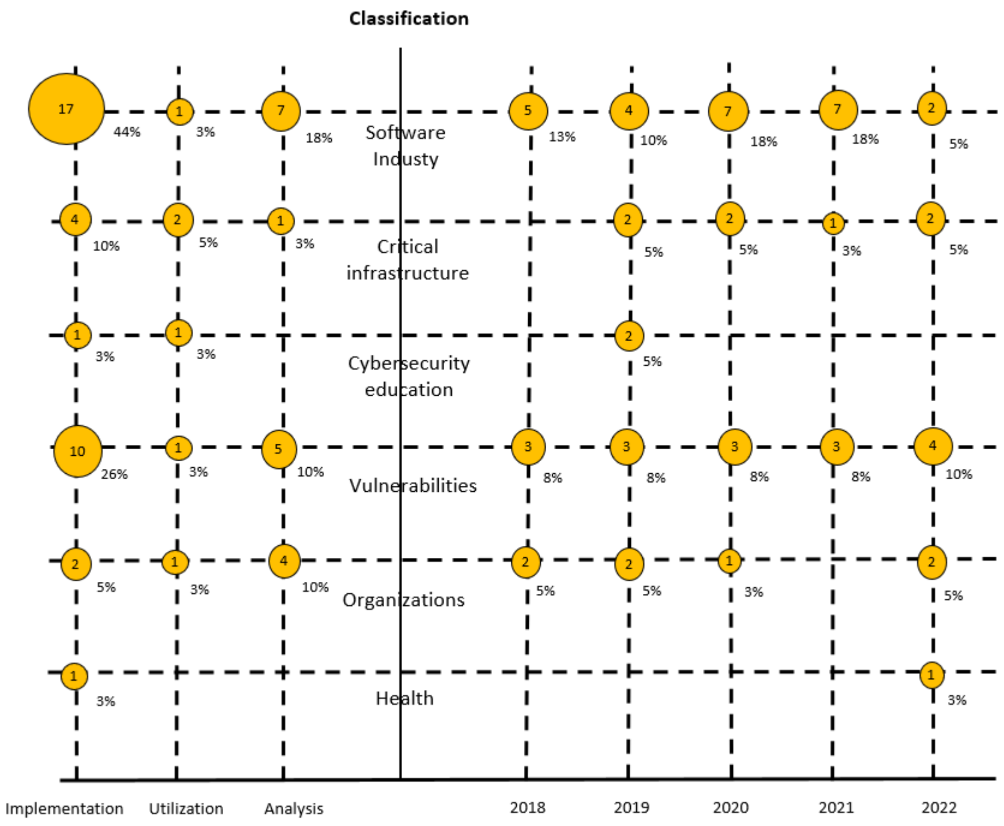

Figure 7. Representation of the systematic mapping.

Of the 39 works selected, 72% (28) correspond to works that generate new proposals regarding developing secure software (implementations). 30% (12) are papers aimed at analysis, literature review, or comparison of studies related to secure software development. Only 10% (4) are works that are based on applying proposals from other authors. Some works are categorized into more than one type of work; that is, a work may present a proposal as well as an analysis of the literature, in which case, we have classified it as "implementation" and "analysis".

6.3. Analysis of Individual Articles

Next, from the articles resulting from the previous phases, we will analyze those with the highest number of citations and that directly contribute to the research questions. They will be categorized according to the following dimensions:

6.3.1. Theoretical Proposals

In the study [68], we highlight the proposal of a framework for aligning software engineering with security engineering, where security activities are proposed directly in each phase of the agile SDLC, for this security engineering activity are mapped into common practices, processes, and techniques of agile software development.

In a recent paper [69], the authors propose a new model for software development that incorporates security and consists of seven levels of security assurance: Governance and Security Threat Analysis, Secure Requirements Analysis, Secure Design, Secure Coding, Secure Test and Review, Secure Implementation, and Security Enhancement. This new proposal recommends various security practices to be followed in each phase of the SDLC to achieve a secure SDLC for global software development (GSD) vendor companies. The results of the case studies indicate that the proposed Software Development model helps to measure an organization's level of security assurance.

Another interesting study [70] relates to a proposal to assess vulnerabilities in software systems through software threat modeling supported by machine learning, reducing the manual intervention of experts. The paper also contributes to the field of information security by helping to reconcile data available across the industry for the benefit of software development teams, focusing primarily on applying data analytics to threat modeling and risk assessment.

6.3.2. Implementation

In [71], a methodology is proposed that promotes security in individual developers' software products. The usefulness of the resulting methodology has been evaluated through a case study conducted in an academic environment. The results successfully show the usefulness of the methodology in building secure and high-quality software applications in the areas of Education, Health, Government, and Business; however, it is important to consider that the academic environment is very different from the industrial one, which makes it difficult to generalize the results of the study.

6.3.3. Review and/or Analysis

In the article [72], the identification of security requirements through the discovery process, the selection of security patterns for the identified security requirements, the design of security requirements using security building blocks, the creation of test templates to support the implementation of patterns during the development stage, vulnerability scanning and secure configurations are key functionalities in the proposed framework. There is a related work [72], the concern of which is to address the challenge of agile software development by addressing Scrum in a framework called FISA-XP. The authors developed this framework to build secure software; however, it is focused on the context of IoT and intelligent transportation systems (ITS).

6.3.4. Uses

In the search, our attention was drawn to the article [73], which provides an overview of four categories or approaches to achieve the security of software systems, static and dynamic analyses, formal methods, and adaptive mechanisms. The authors also intuitively present and demonstrate their applications with several examples, listing the strengths and weaknesses of each approach to help software engineers make informed decisions.

Another interesting study [74] presents a series of modules designed to teach students the fundamental concepts of software engineering from a security perspective: Software Requirements, Software Design, Software Construction, Testing, and Maintaining Secure Software. The evaluation of the educational effectiveness of the modules is still pending, and for which experiments will be conducted with the students. Using pre-and post-tests. The level of understanding of the students in the practice of secure software will be examined. Qualitative data will also be collected and analyzed to look at the perception and analyzed to observe students' perception and awareness of the importance of securing software.

Table A1 shows the list of articles selected for review.

6.4. Response to Research Questions

The main objective of this research is to define a methodological proposal to develop software that integrates the security dimension into all phases of its life cycle.

To achieve this, we defined a research design composed of three phases. The first corresponds to the present work, which seeks to identify all the existing formal literature, noting the strengths and weaknesses of the current proposals. The 39 papers ultimately selected were classified into three dimensions: time, category, and type of work.

Of the 39 selected articles, there are 17 (44%) that propose implementations in the area of the software industry, from which the answers to the defined research questions can be derived.

RQ1: How many articles are related to secure software development?

Mapping obtains data on the main related works about software development that integrate security. Of the 39 works found, we identified 35 (see Figure 8) that address security as an important part of secure software development, offering new proposals regarding secure software development, and among these articles, 17 focus on the software industry according to the classification performed. We made this classification to evaluate the state of security integration in agile software development. The classifications used correspond to three dimensions: by time, by context where they are developed, and type of work.

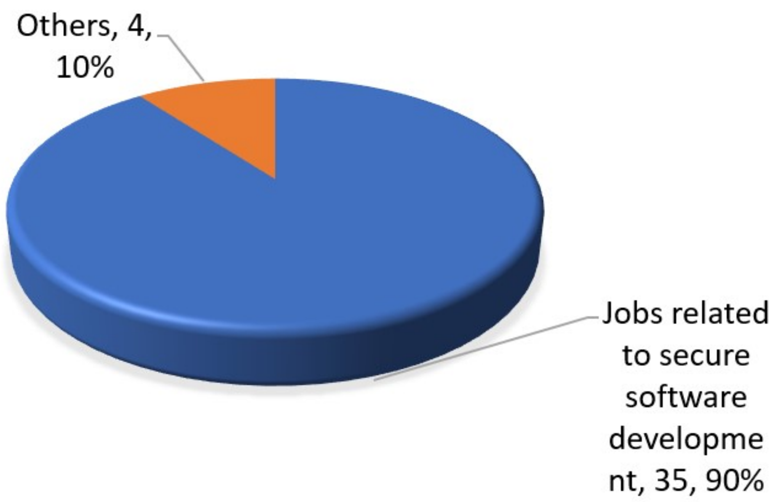

Figure 8. Jobs related to secure software development.

RQ2: How many articles study security practices for agile software development?

We found 5 works that study the integration of security in agile development environments, see Figure 9, of which [68] stands out for establishing security requirements for the design, implementation, verification, and release of secure software through an efficient software security development process. We cannot fail to mention the work reported [71] proposes an agile, secure software development methodology that promotes quality and security in software products of individual developers; however, this proposal does not provide a clear picture of use since it is evaluated in an academic environment and it is, therefore, difficult to generalize the results of the case study.

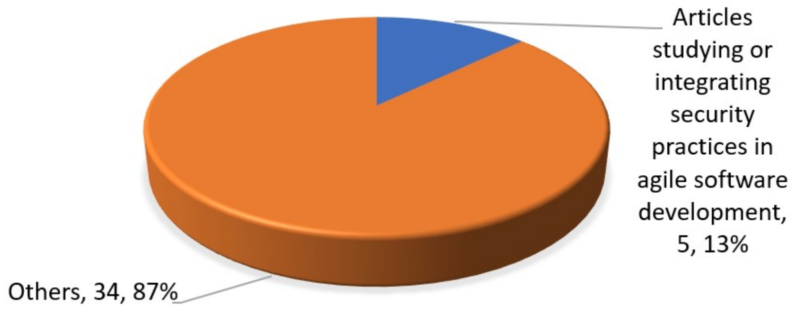

Figure 9. Articles studied on security practices in agile software development.

The resulting three related papers build on [72], a comprehensive and detailed review of agile software development in the context of IoT, ITS, and its cybersecurity and risk challenges. An automated risk assessment approach is explored in [69], and an exploratory study on agile security practices adopted by software developers and security practitioners is presented in [75].

RQ3: What is the context/setting where the articles take place?

Based on Figure 7, it is possible to determine the contexts in which the selected research is conducted. Most of the works are developed in the software industry, with 25 articles. This is followed by 16 articles undertaken in the area of vulnerabilities. In the area of critical infrastructure, we have 7 works as well as in organizations and finally with 2 and 1 work in the areas of cybersecurity education and health, respectively. See Table 4.

Table 4. Contexts of the articles.

Classification	Total
Health	1
Organizations	7
Vulnerabilities	16
Cybersecurity Education	2
Critical Infraestructure	7
Software Security	25

RQ4: What are the SDLC phases that have most addressed security in software development?

We found several articles that propose methods to address security in software development involving all phases of the software development cycle. In this sense, it is essential to point out that, of the 17 articles that propose implementations, there are 9 articles that address security in all phases of the SDLC, which represents 23% of the total.

From the above analysis, we can answer the following research question:

RQ5: Which particular phase of SDLC has been least discussed and addressed in the literature?

Observing Figure 10, we can conclude that the Analysis and Design, Development, and Maintenance phases are the least addressed in terms of security.

RQ6: What results have the new methods or models yielded to ensure security in the development of secure software?

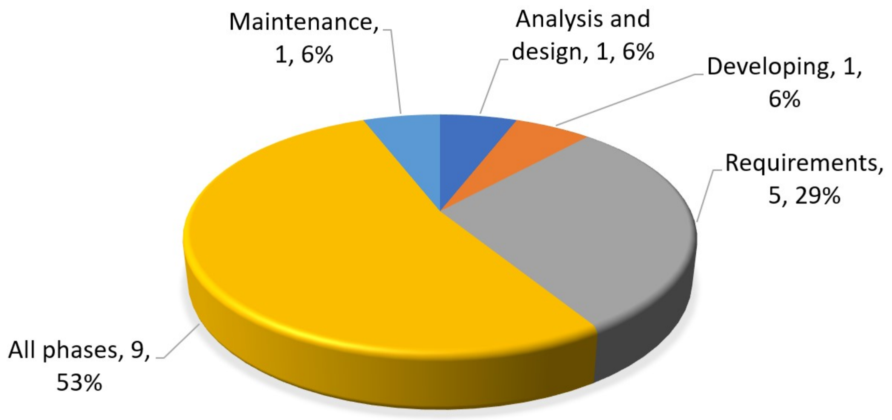

Figure 10. Software development phases that use security in software development.

In the papers analyzed, several results can be distinguished, see Figure 11. According to the findings, 26 articles do not present evaluations to discover if the proposed model is better or worse, which represents 67% of the articles. By contrast, 11 articles evaluate the proposal, which represents 28%.

The results obtained from these evaluations are varied, according to Figure 12: of the selected articles, 10 (26%) do not indicate whether the result is better or worse, 19 indicate that they have good results from their proposed framework (49%), one article (2%) indicates that the result obtained was worse after evaluating the framework, 4 papers (10%) indicate that the result is neutral, and finally there are 5 articles (13%) that do not indicate this result.

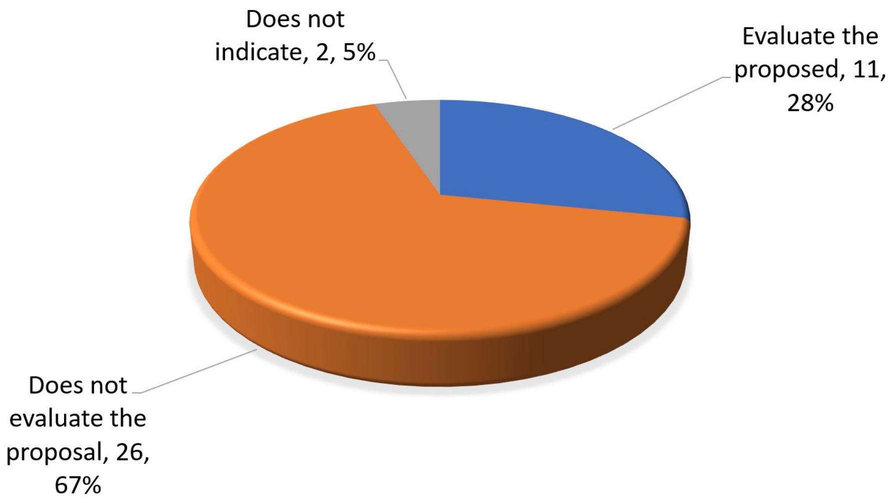

Figure 11. Articles evaluating the proposed model.

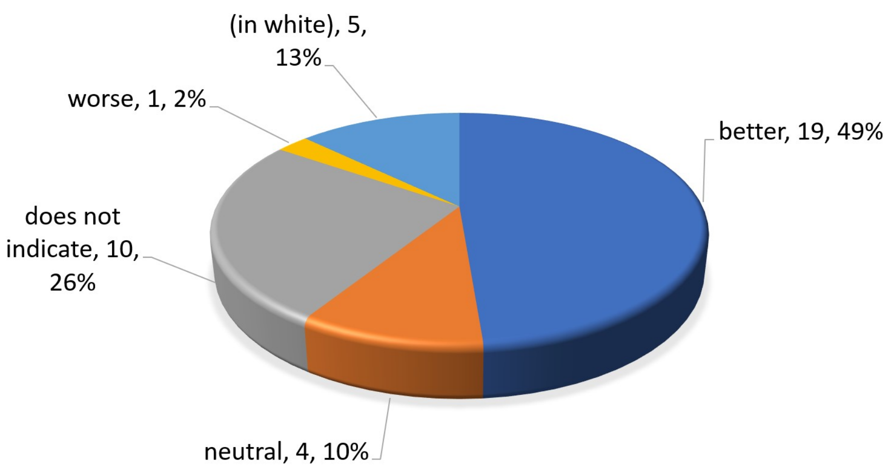

Figure 12. Results obtained.

7. Discussion

7.1. Analysis

Software development teams need a particular method to develop software, which translates into greater competitiveness and success using appropriate practices, tools, and techniques. Today's job market demands more frequent and robust software releases against possible security attacks that may affect them.

Agile methods currently offer a highly flexible and rapid software development process [76]; however, security is neglected in favor of adaptability to new requirements. Several secure software development frameworks, approaches, and maturity models provide some support to developers for incorporating security into agile environments.

A related paper presented by [58] contributes to understanding how teams integrate and model threats in the context of agile development. They also present a list of recommendations that can help companies improve their threat modeling strategies. Engineering programs have yet to address the importance of including security in software design. In this regard, the work proposed by [21] proposes a practical approach to integrate security into the engineering systems development lifecycle and support engineering students in developing secure products and processes. The proposed model is based on an adaptation of SecSDM (Secure Software Development Methodology) [21].

This paper aims to provide an overview of the literature according to [67] to identify approaches or work models applied to secure software development, to define which phases of the software development cycle are the least addressed, and what have been the results of these proposals. To achieve this, we focus on the systematic mapping methodology that allows us to have an overview of a research area through the classification [66] and to share the current trends of existing frameworks for software development involving security.

Our research analyzes the current literature in several ways. First, by time, this temporal dimension classifies the works according to the year of publication based on the assumption that we have selected works from the last five years. The analysis evidence that the highest number of articles is concentrated in 2019 and 2020. There is a decrease in 2021 and 2022, possibly due to the pandemic. The second classification corresponds to the context dimension where the research was conducted: Health, Organizations, Vulnerabilities, Cybersecurity Education, Critical Infrastructure, and Software Industry. According to our research, the most significant number of articles is concentrated in the application category, with 28 (72% of the total). Of these, 17 correspond to the Software Industry category according to the context in which they are developed, representing 44% of the total.

The third dimension corresponds to the type of work in which the article is framed. This classification defined three types: (i) Implementation: corresponding to works that generate new proposals about secure software development. (ii) Analysis: those works that perform analysis or comparisons of studies related to secure software development or when the works correspond to bibliographic reviews, and (iii) Utilization: those works that are based on applying other authors' proposals on topics related to secure software development. Of the 39 selected papers, 72% (28) correspond to papers that generate new proposals related to secure software development (implementations). The 30% (12) are works oriented to the analysis, bibliographical review, or comparison of studies related to secure software development. Only 10% (4) are works based on the implementation of other authors' proposals. Some papers are classified into multiple types of work, i.e., a paper may present both a proposal and a literature analysis. In this case, we have classified it as "implementation" and "analysis". The results show that there is a concern about addressing proposals related to software development. Of the 39 works selected, 72% fall into this category (28). It is observed that slightly more than half of the articles analyzed (54%) address security in one of the phases of the software development cycle (SDLC), corresponding to 21 articles. However, a significant number of articles (26%) refer to the evaluation of security metrics, comparisons, and risk analysis, among others that do not consider applying security in any phase of the SDLC. Finally, eight articles were found that do not consider the use of security in the SDLC phases.

Our results indicate (see Figure 13) a lack of attention to security education, which results in finding engineers with no experience in security practices in software development. In the vulnerability identification and mitigation category, the maintenance and validation stage is the most used (30% of this category). The studies related to the code and requirements level implementation phases are less treated, reaching 20% of the category implementations. There are no related works in the secure modeling and analysis category. The number of related studies corresponds to a study presenting an evaluation of security issues in DevOps. We found two studies that indicate dealing with security in all phases. One of them [77] is concerned with proposing a DevOps approach based on metrics that support managers during the decision-making process, which is far from strengthening the security expertise of young engineers. The second case [78] uses static analysis and software metrics to evaluate the internal security level of software products; however, its results indicate that it only applies to JAVA applications, and the presentation of results is difficult to understand, especially for people with little or no technical knowledge. This has been represented in a pie chart where 10 studies are related to the vulnerability identification and mitigation category, representing 26% of the total number of studies identified.

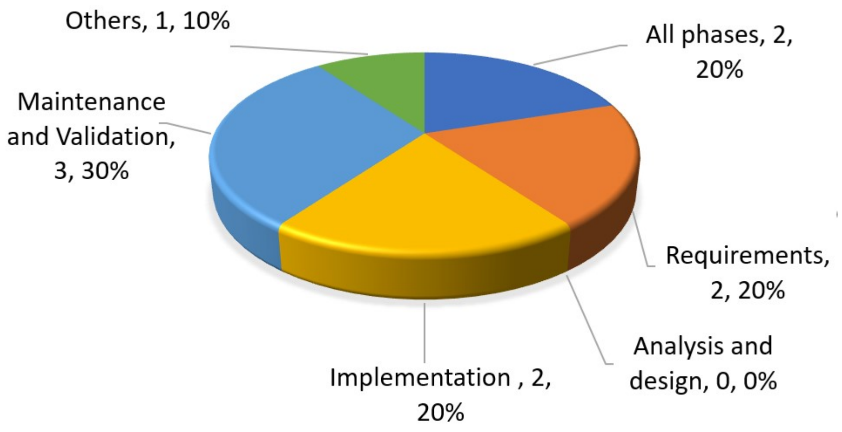

Figure 13. Vulnerabilities classification.

This allows us to provide a scenario for researchers to create a model or framework that covers the main security challenges in the software lifecycle considered to be used by professionals without security expertise. Future work can focus on developing new approaches to improve security in software development without significantly compromising agility gradually.

7.2. Research Gaps

Based on the literature review, we see that the following research gaps exist. Most of the works that we found in the literature, particularly on secure software development, focus on robust work teams and the proposed methods have not been empirically validated by inexperienced development teams.

There is a need to cleverly integrate security practices with agile software development practices without compromising the agility of the resulting product. Agility is key for software as it makes acceptance easier, especially by independent developers. The integration of customer needs with software development processes and security needs are not adequately addressed and requires further exploration. These elements are essential to advance an agile development methodology from a security perspective. All of this requires adequate education by software development educational communities on security practices with agile software development practices.

Considering the above, a method is needed that can integrate security practices into the development of agile software that is focused on professionals, and developers without experience in software development.

8. Proposals

Secure software development must be based on more than the experience and skills of the programmers or team. It must be based on models or frameworks that integrate security practices with agile software development practices. Agility is a key concept that facilitates acceptance by users and, especially, developers. Both elements are essential to moving toward an agile development model that incorporates security as an integral part of the process. Although some works have been identified that address security in the construction of systems, none explicitly point to a model that facilitates the integration of security in developments carried out by developers or teams with no experience in this area.

Given the uncertain environment surrounding software development and the growing threat of cyber attacks with serious consequences, especially when they affect critical systems that handle sensitive information, it is essential to train inexperienced engineers in secure development issues. This will allow them to have a method that integrates security practices throughout the software development process, without affecting agility. To address this problem, it is proposed as future work the creation of a model or proposal that guides the development of secure and agile software, highlighting the benefits of integrating security and agility in a single methodology. This model should focus on the inclusion of security practices from the design phase to the delivery of the software, and on the implementation of agile methodologies that allow a fast and efficient delivery of the software. The implementation of this methodology is expected to reduce software security risks and improve software quality, thus, satisfying the needs of users and customers. In addition, training in secure and agile development practices will allow software engineers to develop skills and knowledge that will be key to facing current and future challenges in secure software development.

Security in software development is a critical issue due to its implications for both users and companies. To address this issue, a systematic mapping of the existing literature on secure development models and practices in the last 5 years was carried out. From the results obtained, research gaps were identified, such as the lack of empirical validation of the proposed methods in inexperienced teams. In addition, it was found that most of the work found focuses on robust teams, which leaves a gap in research on how to apply security practices in less experienced teams. To address the identified research gaps, the next step is to develop a method to integrate security practices into all stages of the software life cycle. This method should be designed with the goal of reducing software security risks and improving software quality. In addition, it should include security training and awareness for students, which will contribute to fostering a security culture in the software development process. It is important to note that this comprehensive and proactive approach to security from the early stages of the software development process will allow early identification of risks and their resolution in a more efficient manner. In this way, the resulting software is expected to be more secure and reliable, satisfying the needs of users and businesses. In summary, the proposal is to develop a method to integrate security in all stages of the software life cycle, improve security awareness and training of software developers, and foster a culture of security in software development.

Figure 14 shows the proposal we plan to carry out, which consists of the following phases:

Phase 1: We adopt a systematic literature review approach to explore the security challenges in web application development and to identify existing security gaps. This review will allow us to comprehensively analyze existing information in the field and establish a solid framework for our work.

Phase 2: To validate the findings obtained from the systematic literature review and to identify possible additional security challenges in web application development, we will conduct a questionnaire survey. This approach will allow us to gather information directly from novice web development professionals and empirically validate security gaps.

Phase 3: In this phase, we will develop the proposed methodology by identifying security controls and agile practices to be combined and integrated from the beginning of the process and in each iteration. The methodology will be implemented and evaluated after each iteration to ensure that the appropriate practices are followed and the established quality and security objectives are met.

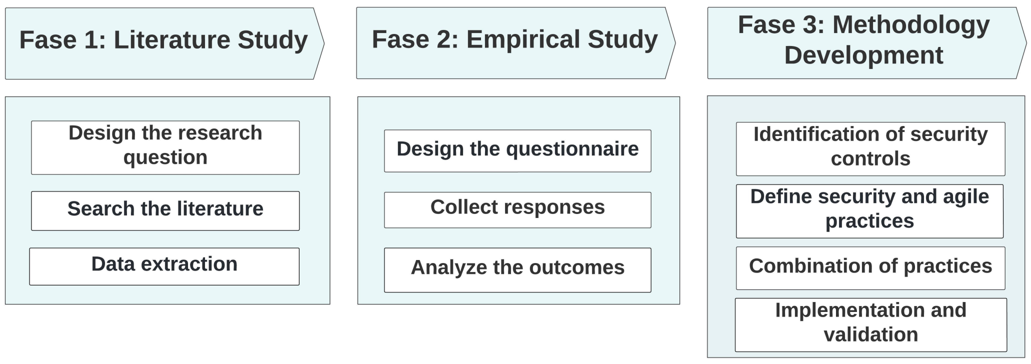

Figure 14. Research Design.

9. Limitations of the Study

We have identified some limitations in the present study that must be taken into account. In our literature search, we found 39 articles published between 2018 and 2022 that generate proposals, analyses, literature reviews, or comparisons of studies related to the development of safe software. We found only 17 that propose implementations of some software development framework or model; however, we found no proposals oriented towards work teams without developing experience that has been validated empirically. We identified four types of threats to the validity of the study [15] which are essential to any type of study since absolute impartiality cannot be guaranteed.

9.1. Construct Validity

Systematic mapping entails structural threats that are relevant to the classification of the studies selected. We generated a search string using the WoS, Scopus, IEEE Xplore, ACM Digital Library, and Science Direct databases. If we take only the search engine statistics into account, then most of the works related to secure software development were found. To avoid important studies being left out, we looked for articles such as scopes of work, security evaluations in systems, static and dynamic analyses after building the software, and proposals for implementations oriented to secure development.

9.2. Internal Validity

The internal validity to conduct the study was measured according to the criterion of two reviewers to identify the classification of the reviewed and selected works, where the exclusion and inclusion criteria were applied. For this, Cohen's kappa index [79] was used for the criteria consolidation.

9.3. External Validity

The study applied a methodology used and accepted by the scientific community, which enabled it to be verified by following the steps presented. Therefore, the results are considered reliable in the study domain.

9.4. Conclusión Validity

There is always the chance of making incorrect connections between the findings and results, which would lead us to draw vague or very possibly erroneous conclusions. To reduce this possibility of error, the data and their relationships were reviewed by more than one investigator, and visual supports were also designed, taking the results as their base.

10. Conclusions

The main focus of this paper is to identify the main research trends in secure software development and to know the results of their proposals. The main findings are summarized as follows: 9810 articles were selected using a search string established from the PICO (Population, Intervention, Comparison, and Outcomes) strategy outlined in [67]. The results were limited to the last five years, from 2018 to 2022, obtaining a total of 368 articles. Repeated articles, as well as those that were already literature reviews, were eliminated from the selection so that they would not influence or affect the present study. With this limitation of articles, the total number of papers selected for the first reading was 312 articles. After reading the abstract, 51 articles were selected for full reading, of which 12 were not included in the study because we did not have access to them. Finally, the study was carried out with 39 scientific articles.

The 39 articles selected and reviewed were classified into three dimensions for analysis: by time, by context in which they are developed, and by type of work. The time dimension classifies the works according to the year of publication, considering the limitation that we have considered of the last 5 years. The context dimension corresponds to the scenarios where the research is developed; the scenarios considered correspond to: Health, Organizations, Vulnerabilities, Cybersecurity, Education, Critical Infrastructure, and Software Industry. With respect to the categories according to the type of work, the dimensions were we have the category Implementation, which corresponds to works that generate new proposals for the development of secure software. Analysis categories are those works that perform analysis or comparisons of studies analysis or comparisons of studies related to secure software development, or when the works correspond to reviews of Finally, the Use category corresponds to those works that are based on applying or presenting proposals from other authors on topics related to secure software development.

The analysis shows that the largest number of works is concentrated in the category of implementations, with 28 works representing 72%. Of these 28 studies, 17 correspond to the Software Industry category according to the context in which they are developed, representing 44% of the total. Regarding the time dimension of these 17 works, it is evident that their concentration is in the year 2020 with 6 works, then there is a decline in the years 2021 and 2022, which may be due to the pandemic. According to our research, it is essential to note that of the 17 articles that propose implementations, there are 9 articles that address security in all phases of the SDLC, which represents 23% of the total selected.

From the above analysis, we can conclude that the Design, Implementation, and Validation phases are the least addressed in the analyzed literature. In response to the results obtained from the new development models or frameworks to ensure security in software development, several results are distinguished; according to the findings, there are 10 articles that do not indicate whether the result is better or worse, which represents 26% of the articles, 19 articles indicate having good results of their proposed implementations with 49%, only 1 work indicates that its result was worse after the evaluation (2%) and 4 works indicate that the result is neutral with 10%.

The contribution of this paper is to share the current trends about the existing methods for software development involving security. With this scenario, it is possible to provide resources to researchers that allow them to come up with a general framework covering the main security challenges in the software development life cycle, which aims to drive the development of more secure software by inexperienced developers.

As future work, it is expected that this study will be an input for further research that seeks the construction of a method that is aware of security throughout the software construction process and that is focused on inexperienced teams or developers. It is also expected to guide and be a reference for future research in the field of software engineering, considering the limitations found. Another proposal for future work is the search for works that contemplate security in mobile developments and that can provide a basis to contribute to these new technologies, secure and more reliable developments.

Author Contributions: Y.V.-R. contributed to the writing and development of the manuscript in all its phases. J.H.-D. contributed to the methodology used and discussion. J.D.-A. contributed to the analysis of the results and limitations of the study. R.C.-M. contributed to the review of articles. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by Proyecto DIUFRO DI22-0043 Universidad de La Frontera, Temuco, Chile. Jaime Díaz-Arancibia is supported by Grant ANID, Chile, Fondecyt de Iniciación en investigación, Project No 11230141.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable

Acknowledgments: The authors wish to thank the Universidad Americana de Europa and its Doctoral Program in Informatics. More, special thanks to the scientific research initiative DIUFRO, funded by the Universidad de La Frontera, project DI22-0043, and Grant ANID, Project No 11230141.

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

Abbreviations

The following abbreviations are used in this manuscript:

SDLC	Systems Development Life Cycle

Appendix A

Table A1. Articles using security in SDLC phases.

Number	Article	Cite	Brief Description
1	Capturing Software Security Practices using CBR: Three Case Studies	[80]	The original goal is to investigate how to change the software engineering team's attitude towards security and how to help them practice security throughout the software development lifecycle.
2	Engineering Security Vulnerability Prevention, Detection, and Response	[81]	This article indicates that providing tools to help with software security is not enough. Students and practitioners need to be trained on the importance and practices for designing and developing secure systems.
3	A Framework for Teaching Security Design Analysis Using Case Studies and the Hybrid Flipped Classroom	[82]	The first part of this article examines the security design analysis techniques described in the literature. The second part of this section focuses on the methods used to teach software development.

Table A1. *Cont.*

Number	Article	Cite	Brief Description
4	A hierarchical model for quantifying software security based on static analysis alerts and software metrics	[78]	The proposed SAM model is based exclusively on static analysis. This allows the SAM model to be applied regularly during the software development process. Software from the earliest stages of implementation in a fully automated manner, as static analysis does not require the running of the software product being tested
5	What are the critical security flaws in my system?	[83]	A static analysis is proposed to determine the severity of a vulnerability
6	Hazard Analysis Methods for Software Safety Requirements Engineering	[84]	The document proposes and reviews three different hazard analysis methods (STPA, FHA, and SFMEA) used in software requirements engineering to develop software security requirements. Each method has been used in practice in various security programs.
7	Aligning security objectives with agile software development	[68]	A framework is proposed to align software engineering with security engineering. For them, it proposes security activities directly in each phase of the agile SDLC.
8	Closing the Feedback Loop Between UX Design, Software Development, Security Engineering, and Operations	[85]	The new TDLC model introduces one more circle in the infinite loop where the four phases of the model are included at the beginning DoubleDiamond. The design phases lead to the development phases, then to the operations phases, and then back to the beginning.
9	An Empirical Investigation of Agile Information Systems Development for Cybersecurity	[75]	In this exploratory study, we empirically explore agile security practices adopted by software developers and security professionals.
10	Secure SDLC Using Security Patterns 2.0	[86]	The proposed framework integrates security concerns from the early stage to the removal stage and, thus, software security vulnerabilities are found and mitigated at early stages of SDLC and save a lot of re-engineering costs for vulnerabilities. post implementation.
11	Validation of the smbc framework of security testing using analytic hierarchy process	[87]	The SMBC framework addresses the issue of security testing in the design phase of SDLC
12	Security requirements specification: A formal method perspective	[88]	In this article, a framework for the specification of security requirements by formal methods is proposed. The goal of the proposed framework is to specify security requirements formally and integrate them with SDLC
13	Secure modules for undergraduate software engineering courses	[74]	This document presents a series of modules that are designed to be integrated into undergraduate software engineering courses from a security perspective. The objective of the modules is to teach the creation of strong software security requirements, design, and development of secure software, and verification of secure software through a secure software development life cycle.
14	Security and software engineering	[73]	In this article, we first provide an introduction to the principles and concepts of software security from a software engineering point of view. We then provide an overview of four categories of approaches to achieving security in software systems, namely static and dynamic analysis, formal methods, and adaptive mechanisms.
15	MDA approach for application security integration with automatic code generation from communication diagram	[89]	In this work, a new contribution to the generation of secure applications is proposed with its security mechanisms based on the MDA approach to address the functional and non-functional aspects during the software engineering process.
16	Automated Risk Management based Software Security Vulnerabilities Management	[70]	This work presents a quantitative threat modeling as part of a comprehensive software security management system
17	Security Assurance Model of Software Development for Global Software Development Vendors	[69]	Proposed model to be used by global software developer (GSD) providers. The evaluation process of this model called Software Development SAM is based on the Motorola evaluation tool.

Table A1. *Cont.*

Number	Article	Cite	Brief Description
18	A Preventive Secure Software Development Model for a Software Factory: A Case Study	[34]	This work proposes the Emerging Secure Software Development Model called Viewnext-Uex, preventive and flexible. His findings after being empirically evaluated show that methodologically improves software security with the application of the proposed model. The security and quality of the software are increased, as well as development productivity.
19	A Readiness Model for Security Requirements Engineering	[90]	The goal of this document is to develop a Security Requirements Engineering Readiness Model (SRERM) to enable organizations to assess their security requirements engineering readiness levels.
20	A Novel Lightweight Solo Software Development Methodology With Optimum Security Practices	[71]	The purpose of this document is to introduce the Secure-SSDM model to individual software developers. The study has successfully demonstrated the usefulness of Secure-SSDM in building high-quality and secure software applications in the fields of Education, Health, Government, and Business.
21	Agile Approaches for Cybersecurity Systems, IoT and Intelligent Transportation	[72]	This document presents a complete and detailed review of agile software development in the context of IoT, ITS, and its cybersecurity and risk challenges.
22	A Hybrid Model of Hesitant Fuzzy Decision-Making Analysis for Estimating Usable-Security of Software	[91]	The main objective of this work is to evaluate the safety of use of a software that focuses on its two features. The evaluation of usable security would also be helpful in improving security and ease of use for end-user satisfaction and privacy.
23	Fuzzy Expert System of Information Security Risk Assessment on the Example of Analysis Learning Management Systems	[92]	Proposes a new hierarchical structured model for information security risk assessment using fuzzy logic. This methodology can be used to assess the information security risks of any complex automated management system (socially significant ERP system) used in other areas.
24	Text Categorization Approach for Secure Design Pattern Selection Using Software Requirement Specification	[93]	This paper proposes to use a repository of secure design patterns as a dataset and a repository of requirements artifacts in the form of a software requirements specification (SRS).
25	Prioritization Based Taxonomy of DevOps Security Challenges Using PROMETHEE	[94]	The objective of this study is to identify and develop a taxonomy based on the prioritization of DevOps security challenges. A total of 18 DevOps security challenges were extracted using a systematic literature review approach and further evaluated with experts using a questionnaire survey study. Finally, the PROMETHEE-II multi-criteria decision-making approach was used to prioritize and develop the taxonomy of the identified factors and their categories.
26	A Crisis Situations Decision-Making Systems Software Development Process With Rescue Experiences	[95]	In this article, a customized version of XP called Crisis Decision Systems Software Development Process (CSDP) is proposed. CSDP is the result of the authors' experiences while participating in the rescue agent simulation division of the RoboCup competitions from 2006 to 2010. CSDP is an agile and fast process that makes the development team able to respond to changes suddenly in the shortest possible time.
27	Automatic Classification Method for Software Vulnerability Based on Deep Neural Network	[96]	In this article, a new automatic vulnerability classification model called TFI-DNN has been proposed. To better analyze and manage vulnerabilities according to their membership classes, improve system security performance, and reduce the risk of the system being attacked and damaged, this paper applied a deep neural network to vulnerability classification. of software. The results show that the proposed TFI-DNN model outperforms others in accuracy, accuracy, and score and works well on the rate of Recovery.

Table A1. *Cont.*

Number	Article	Cite	Brief Description
28	Reusable Security Requirements Repository Implementation Based on Application/System Components	[97]	In this article, a repository model has been proposed that addresses the issue of reusing security requirements. The repository has a structure that guides the user on what type of information should be reused. Flexibility is an advantage of the proposed model. The model allows the definition of requirements at any level of precision. The proposed model does not include risk factors, risk analysis, and risk management. This activity has not been included in the model on purpose to achieve simplicity and allow the model to be used in conjunction with existing risk analysis techniques in an organization.
29	GMSA: Gathering Multiple Signatures Approach to Defend Against Code Injection Attacks	[98]	In this paper, we introduce a tool called GMSA, developed to detect a variety of CIA, for example, Cross-Site Scripting (XSS) attack, SQL injection attack, shell injection attack (command injection attack) and file inclusion attack. The latter consists of local file inclusion and remote file inclusion.
30	A Survey on Blockchain Acquainted Software Requirements Engineering: Model, Opportunities, Challenges, and Future Directions	[99]	In this article, we provide a novel comprehensive review of blockchain-related aspects of SRE requirements engineering practices. We introduce SRE-based quality improvement factors and describe the need for blockchain technology in this domain. Additionally, they have classified SRE practices based on blockchain engineering.
31	Integrating Model Checking With SysML in Complex System Safety Analysis	[100]	In this paper here, we propose the integration of model checking with the systems modeling language to analyze the security of complex systems. Systems Modeling Language (SysML) is introduced to establish a unified system model that can describe a hybrid system of hardware and software but cannot be directly applied to security analysis. Using SysML makes it easy for designers, analysts, and vendors to use the unified model. The semi-formal SysML model is then transformed into the formal NuSMV model, which is used to perform security analysis and verification.
32	A Novel Key Agreement Protocol Based on RET Gadget Chains for Preventing Reused Code Attacks	[101]	This paper proposes a new key agreement protocol based on the RET device chain. The novel protocol not only considers cryptographic security techniques and control flow integrity when executing programs, but it can also prevent vulnerability attacks during implementation at the source code level.
33	An Evaluation of Quantitative Non-Functional Requirements Assurance Using ArchiMate	[102]	This document introduces a system architecture assessment method that can perform a quantitative NFR assurance assessment for the system architecture through ArchiMate. The document also proposes an algorithm to automate the quantitative evaluation process. A questionnaire survey among software engineers and a case study on a vehicle safety monitoring system were conducted to verify the necessity of the method. Additionally, an experimental design with 18 samples divided into 2 groups was presented to compare how the independent variables affect the dependent variables. The results of the experiment demonstrate that the proposed method achieves a better NFR evaluation effect than the traditional approach. The proposed method is expected to be used in the early stage of software development projects for system NFR development, such as requirements analysis, system architecture design, and system modeling.
34	Self-Service Cybersecurity Monitoring as Enabler for DevSecOps	[103]	This document focuses on self-service cybersecurity monitoring as an enabler to introduce security practices in a DevOps environment. The case study provides evidence of how this cybersecurity monitoring infrastructure enabled threats to be detected, such as denial attacks, and helped better anticipate phishing problems.

Table A1. *Cont.*

Number	Article	Cite	Brief Description
35	Automatically Patching Vulnerabilities of Binary Programs via Code Transfer From Correct Versions	[104]	This document presents BINPATCH, an algorithm for automatically patching known vulnerabilities in binary programs. It first locates the faulty function, which contains the vulnerability, through a comparison of similar codes. It then reuses the corresponding code from the correct version of the faulty function as patch code and inserts it into the faulty function using binary rewrite. BINPATCH is tested on eight real-world vulnerabilities, and experimental results show that it is capable of not only locating faulty code effectively but also patching code correctly.
36	Classifying Software Vulnerabilities by Using the Bugs Framework	[105]	In this paper, data mining techniques are used to identify software vulnerabilities, classify them into different categories using the bug framework proposed by the National Institute of Standards and Technology (NIST) and design a model to predict the weakness of future vulnerabilities.
37	Metrics-driven devsecops	[77]	In this paper, a unique metrics-based approach is proposed to help improve software engineering processes by increasing software quality, adaptability, and security, and decreasing costs, and time to market.
38	The Impact of Software Security Practices on Development Effort: An Initial Survey	[106]	The objective of this study is to obtain an overview of the application of software security practices in the industry and to identify the impact of introducing such activities on software development projects in terms of effort/cost.
39	Context-Sensitive Case-Based Software Security Management System	[107]	In this paper, we highlight the need to include application context-sensitive modeling within the case-based software security management system proposed by the authors. This article expands on previous work to include application context modeling. The proposed idea builds software security models using an application context.

References

1. Faheem, M.; Shah, S.B.H.; Butt, R.A.; Raza, B.; Anwar, M.; Ashraf, M.W.; Ngadi, M.A.; Gungor, V.C. Smart grid communication and information technologies in the perspective of Industry 4.0: Opportunities and challenges. *Comput. Sci. Rev.* **2018**, *30*, 1–30. [CrossRef]
2. Lee, M.; Yun, J.J.; Pyka, A.; Won, D.; Kodama, F.; Schiuma, G.; Park, H.; Jeon, J.; Park, K.; Jung, K.; et al. How to respond to the fourth industrial revolution, or the second information technology revolution? Dynamic new combinations between technology, market, and society through open innovation. *J. Open Innov. Technol. Mark. Complex.* **2018**, *4*, 21. [CrossRef]
3. Liou, J.C.; Duclervil, S.R. A survey on the effectiveness of the secure software development life cycle models. In *Innovations in Cybersecurity Education*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 213–229.
4. McGraw, G. From the ground up: The DIMACS software security workshop. *Secur. Privacy IEEE* **2003**, *1*, 59–66. [CrossRef]
5. Castellaro, M.; Romaniz, S.; Ramos, J.C.; Feck, C.; Gaspoz, I. Aplicar el Modelo de Amenazas para incluir la Seguridad en el Modelado de Sistemas. In Proceedings of the V Congreso Iberoamericano de Seguridad Informática—CIBSI, Bogota, Colombia, 22–24 January 2016; Volume 16.
6. Hernández Yeja, A.; Porven Rubier, J. Procedimiento para la seguridad del proceso de despliegue de aplicaciones web. *Rev. Cuba. Cienc. Inform.* **2016**, *10*, 42–56.
7. Pecka, N.S. Making Secure Software Insecure without Changing Its Code: The Possibilities and Impacts of Attacks on the DevOps Pipeline. Ph.D. Thesis, Iowa State University, Ames, IA, USA, 2022.
8. Konstantinidou, C.A.; Lang, W.; Papadopoulos, A.M.; Santamouris, M. Life cycle and life cycle cost implications of integrated phase change materials in office buildings. *Int. J. Energy Res.* **2019**, *43*, 150–166. [CrossRef]
9. Symantec. Symantec. Internet Security Threat Report. Available online: https://www.symantec.com/security-center/threatreport (accessed on 23 February 2023).
10. Diéguez, M.; Cares, C. Anticipation models (anti-models) for a proactive cyber defence. In Proceedings of the IX Congreso Internacional de Computación y Telecomunicaciones, Lima, Peru, 11–13 October 2017; pp. 247–254.
11. ISO. ISO/IEC27001. Information Security Management. Available online: https://www.iso.org/standard/82875.html (accessed on 23 February 2023).
12. ISO. NIST, Cybersecurity. Available online: http://www.iso.org/iso/catalogue_detail?csnumber=54533 (accessed on 20 February 2023).
13. ISACA. Control Objectives for Information and Related Technologies (Cobit). Available online: http://www.isaca.org/KnowledgeCenter/cobit/Pages/Products.aspx (accessed on 21 February 2023).

14. Ključnikov, A.; Mura, L.; Sklenár, D. Information security management in SMEs: Factors of success. *Entrep. Sustain. Issues* **2019**, *6*, 2081. [CrossRef]
15. Meridji, K.; Al-Sarayreh, K.T.; Abran, A.; Trudel, S. System security requirements: A framework for early identification, specification and measurement of related software requirements. *Comput. Stand. Interfaces* **2019**, *66*, 103346. [CrossRef]
16. Ansari, M.T.J.; Pandey, D.; Alenezi, M. STORE: Security threat oriented requirements engineering methodology. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 191–203. [CrossRef]
17. Mishra, N.; Pandya, S. Internet of things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review. *IEEE Access* **2021**, *9*, 59353–59377. [CrossRef]
18. López-Rodríguez, S.A.; García-Peña, V.R. Metodologías de desarrollo de software seguro con propiedades agiles. *Polo Conoc.* **2021**, *5*, 1027–1046.
19. Filus, K.; Domańska, J. Software vulnerabilities in TensorFlow-based deep learning applications. *Comput. Secur.* **2023**, *124*, 102948. [CrossRef]
20. Kumar, R.; Goyal, R. On cloud security requirements, threats, vulnerabilities and countermeasures: A survey. *Comput. Sci. Rev.* **2019**, *33*, 1–48. [CrossRef]
21. Von Solms, S.; Futcher, L.A. Adaption of a secure software development methodology for secure engineering design. *IEEE Access* **2020**, *8*, 125630–125637. [CrossRef]
22. García-Peñalvo, F. *Proyecto Docente e Investigador. Catedrático de Universidad. Perfil Docente: Ingeniería del Software y Gobierno de Tecnologías de la Información. Perfil Investigador: Tecnologías del Aprendizaje. Área de Ciencia de la Computación e Inteligencia Artificia*l; Technical Report; Grupo GRIAL: Salamanca, Spain, 2018.
23. De Vicente Mohino, J.; Bermejo Higuera, J.; Bermejo Higuera, J.R.; Sicilia Montalvo, J.A. The application of a new secure software development life cycle (S-SDLC) with agile methodologies. *Electronics* **2019**, *8*, 1218. [CrossRef]
24. Hudaib, A.; AlShraideh, M.; Surakhi, O.; Khanafseh, M. A survey on design methods for secure software development. *Int. J. Comput. Technol.* **2017**, *16*, 7047–7064.
25. Ramirez, A.; Aiello, A.; Lincke, S.J. A survey and comparison of secure software development standards. In Proceedings of the 2020 13th CMI Conference on Cybersecurity and Privacy (CMI)—Digital Transformation-Potentials and Challenges (51275), Copenhagen, Denmark, 26–27 November 2020; pp. 1–6.
26. Rindell, K.; Hyrynsalmi, S.; Leppänen, V. Fitting security into agile software development. In *Research Anthology on Recent Trends, Tools, and Implications of Computer Programming*; IGI Global: Hershey, PA, USA, 2021; pp. 1026–1045.
27. McGraw, G. Security Software Building Security in Seven Touchpoints for Software Security. 2023. Available online: http://www.swsec.com/resources/touchpoints/ (accessed on 22 February 2023).
28. Sinha, A.; Das, P. Agile methodology vs. traditional waterfall SDLC: A case study on quality assurance process in software industry. In Proceedings of the 2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), Kolkata, India, 4–5 May 2021; pp. 1–4.
29. Futcher, L.; von Solms, R. SecSDM: A usable tool to support IT undergraduate students in secure software development. In Proceedings of the HAISA, Crete, Greece, 6–8 June 2012; pp. 86–96.
30. Fowler, M.; Highsmith, J. The agile manifesto. *Softw. Dev.* **2001**, *9*, 28–35.
31. Croxford, M.; Chapman, R. Correctness by construction: A manifesto for high-integrity software. *J. Def. Soft. Eng.* **2005**, 5–8.
32. Abundis, C.J.B. Metodologías para desarrollar software seguro. *Recibe. Rev. Electron. Comput. Inform. Biomed. Electron.* **2013**, *3*, 1–6.
33. Lindo, A.C. AC Modelos de Desarrollo Seguro del Software. 2023. Available online: https://web.fdi.ucm.es/posgrado/conferencias/AndresCaroLindo-slides.pdf (accessed on 23 February 2023).
34. Núñez, J.C.S.; Lindo, A.C.; Rodríguez, P.G. A preventive secure software development model for a software factory: A case study. *IEEE Access* **2020**, *8*, 77653–77665. [CrossRef]
35. Microsoft. SDL—Agile Requirements. 2023. Available online: https://learn.microsoft.com/en-us/previous-versions/windows/desktop/ee790620(v=msdn.10)?redirectedfrom=MSDN (accessed on 27 February 2023).
36. BSIMM. BSIMM Frameworks. 2023. Available online: https://www.bsimm.com/ (accessed on 27 February 2023).
37. Chechik, M.; Salay, R.; Viger, T.; Kokaly, S.; Rahimi, M. Software assurance in an uncertain world. In Proceedings of the Fundamental Approaches to Software Engineering: 22nd International Conference, FASE 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, 6–11 April 2019; pp. 3–21.
38. Tawalbeh, L.; Muheidat, F.; Tawalbeh, M.; Quwaider, M. IoT Privacy and security: Challenges and solutions. *Appl. Sci.* **2020**, *10*, 4102. [CrossRef]
39. Beznosov, K.; Kruchten, P. Towards agile security assurance. In Proceedings of the 2004 Workshop on New Security Paradigms, Virtual, 20–23 September 2004; pp. 47–54.
40. Tøndel, I.A.; Jaatun, M.G.; Cruzes, D.S.; Williams, L. Collaborative security risk estimation in agile software development. *Inf. Comput. Secur.* **2019**, *27*, 508–535. [CrossRef]
41. Oueslati, H.; Rahman, M.M.; ben Othmane, L. Literature review of the challenges of developing secure software using the agile approach. In Proceedings of the 2015 10th International Conference on Availability, Reliability and Security, Toulouse, France, 24–28 August 2015; pp. 540–547.

42. Bhasin, S. Quality assurance in agile: A study towards achieving excellence. In Proceedings of the 2012 Agile India, Bengaluru, India, 17–19 February 2012; pp. 64–67.
43. Newton, N.; Anslow, C.; Drechsler, A. Information security in agile software development projects: A critical success factor perspective. In Proceedings of the 27th European Conference on Information Systems (ECIS), Uppsala, Sweden, 8–14 June 2019.
44. 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]
45. Kramer, J.D. Developmental test and requirements: Best practices of successful information systems using agile methods. *Def. AR J.* **2019**, *26*, 128–150.
46. Villamizar, H.; Kalinowski, M.; Garcia, A.; Mendez, D. An efficient approach for reviewing security-related aspects in agile requirements specifications of web applications. *Requir. Eng.* **2020**, *25*, 439–468. [CrossRef]
47. Sharma, A.; Bawa, R. Identification and integration of security activities for secure agile development. *Int. J. Inf. Technol.* **2020**, *14*, 1117–1130. [CrossRef]
48. Bodden, E. State of the systems security. In Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, New York, NY, USA, 27 May–3 June 2018; pp. 550–551.
49. Ancán Bastías, O.; Díaz, J.; López Fenner, J. Exploring the Intersection between Software Maintenance and Machine Learning—A Systematic Mapping Study. *Appl. Sci.* **2023**, *13*, 1710. [CrossRef]
50. Astías, O.A.; Díaz, J.; Rodríguez, C.O. Evaluation of critical thinking in online software engineering teaching: A systematic mapping study. *IEEE Access* **2021**, *9*, 167015–167026.
51. Alenezi, M.; Agrawal, A.; Kumar, R.; Khan, R.A. Evaluating performance of Web application security through a fuzzy based hybrid multi-criteria decision-making approach: Design tactics perspective. *IEEE Access* **2020**, *8*, 25543–25556. [CrossRef]
52. Fernandez, E.B.; Astudillo, H.; Pedraza-García, G. Revisiting architectural tactics for security. In Proceedings of the Software Architecture: 9th European Conference, ECSA 2015, Dubrovnik/Cavtat, Croatia, 7–11 September 2015; pp. 55–69.
53. Abeyrathna, A.; Samarage, C.; Dahanayake, B.; Wijesiriwardana, C.; Wimalaratne, P. A security specific knowledge modelling approach for secure software engineering. *J. Natl. Sci. Found. Sri Lanka* **2020**, *48*, 93–98. [CrossRef]
54. Nguyen-Duc, A.; Do, M.V.; Hong, Q.L.; Khac, K.N.; Quang, A.N. On the adoption of static analysis for software security assessment–A case study of an open-source e-government project. *Comput. Secur.* **2021**, *111*, 102470. [CrossRef]
55. Croft, R.; Xie, Y.; Zahedi, M.; Babar, M.A.; Treude, C. An empirical study of developers' discussions about security challenges of different programming languages. *Empir. Softw. Eng.* **2022**, *27*, 1–52. [CrossRef]
56. Antal, G.; Keleti, M.; Hegedűs, P. Exploring the security awareness of the python and javascript open source communities. In Proceedings of the 17th International Conference on Mining Software Repositories, Seoul, Republic of Korea, 29–30 June 2020; pp. 16–20.
57. Correa, R.; Bermejo Higuera, J.R.; Higuera, J.B.; Sicilia Montalvo, J.A.; Rubio, M.S.; Magreñán, Á.A. Hybrid Security Assessment Methodology for Web Applications. *Comput. Model. Eng. Sci.* **2021**, *126*, 89–124.
58. Bernsmed, K.; Cruzes, D.S.; Jaatun, M.G.; Iovan, M. Adopting threat modelling in agile software development projects. *J. Syst. Softw.* **2022**, *183*, 111090. [CrossRef]
59. Villamizar, H.; Kalinowski, M.; Viana, M.; Fernández, D.M. A systematic mapping study on security in agile requirements engineering. In Proceedings of the 2018 44th Euromicro conference on software engineering and advanced applications (SEAA), Prague, Czech Republic, 29–31 August 2018; pp. 454–461.
60. Weir, C.; Becker, I.; Noble, J.; Blair, L.; Sasse, M.A.; Rashid, A. Interventions for long-term software security: Creating a lightweight program of assurance techniques for developers. *Software: Pract. Exp.* **2020**, *50*, 275–298. [CrossRef]
61. Butler, N. Security in Agile Software Development: A Simple Guide: Bigger Impact. 2022. Available online: https://www.boost.co.nz/blog/2022/02/security-in-agile-software-development#who-the-guide-is-for (accessed on 27 February 2023).
62. Veracode. Agile Security. 2023. Available online: https://www.boost.co.nz/blog/2022/02/security-in-agile-software-development#who-the-guide-is-for (accessed on 27 February 2023).
63. Security, L. 10 Agile Software Development Security Concerns You Need to Know. 2023. Available online: https://www.legitsecurity.com/blog/10-agile-software-development-security-concerns-you-need-to-know (accessed on 27 February 2023).
64. OWASP. OWASP Top Ten. 2023. Available online: https://owasp.org/www-project-top-ten/ (accessed on 27 February 2023).
65. SANS. Web Application Security Awareness Training. 2023. Available online: https://www.sans.org/security-awareness-training/products/specialized-training/developer/?msc=ssa-main-nav (accessed on 27 February 2023).
66. Moher, D.; Liberati, A.; Tetzlaff, J.; Altman, D.G.; PRISMA Group. Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. *Ann. Intern. Med.* **2009**, *151*, 264–269. [CrossRef]
67. Petersen, K.; Vakkalanka, S.; Kuzniarz, L. Guidelines for conducting systematic mapping studies in software engineering: An update. *Inf. Softw. Technol.* **2015**, *64*, 1–18. [CrossRef]
68. Rindell, K.; Hyrynsalmi, S.; Leppänen, V. Aligning Security Objectives With Agile Software Development. In Proceedings of the 19th International Conference on Agile Software Development: Companion, Porto, Portugal, 21–25 May 2018; pp. 1–9. [CrossRef]
69. Khan, R.A.; Khan, S.U.; Alzahrani, M.; Ilyas, M. Security Assurance Model of Software Development for Global Software Development Vendors. *IEEE Access* **2022**, *10*, 58458–58487. [CrossRef]
70. Althar, R.R.; Samanta, D.; Kaur, M.; Singh, D.; Lee, H.N. Automated Risk Management Based Software Security Vulnerabilities Management. *IEEE Access* **2022**, *10*, 90597–90608. [CrossRef]

71. Moyo, S.; Mnkandla, E. A novel lightweight solo software development methodology with optimum security practices. *IEEE Access* **2020**, *8*, 33735–33747. [CrossRef]
72. Tashtoush, Y.M.; Darweesh, D.A.; Husari, G.; Darwish, O.A.; Darwish, Y.; Issa, L.B.; Ashqar, H.I. Agile Approaches for Cybersecurity Systems, IoT and Intelligent Transportation. *IEEE Access* **2021**, *10*, 1360–1375. [CrossRef]
73. Malek, S.; Bagheri, H.; Garcia, J.; Sadeghi, A. Security and software engineering. In *Handbook of Software Engineering*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 445–489.
74. Yang, J.; Lodgher, A.; Lee, Y. Secure modules for undergraduate software engineering courses. In Proceedings of the 2018 IEEE Frontiers in Education Conference (FIE), San Jose, CA, USA, 3–6 October 2018; pp. 1–5.
75. Ardo, A.A.; Bass, J.M.; Gaber, T. An empirical investigation of agile information systems development for cybersecurity. In Proceedings of the European, Mediterranean, and Middle Eastern Conference on Information Systems, Dubai, United Arab Emirates, 25–26 November 2021; pp. 567–581.
76. 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]
77. Mallouli, W.; Cavalli, A.R.; Bagnato, A.; De Oca, E.M. Metrics-driven DevSecOps. In Proceedings of the ICSOFT, Paris, France, 7–9 July 2020; pp. 228–233.
78. Siavvas, M.; Kehagias, D.; Tzovaras, D.; Gelenbe, E. A hierarchical model for quantifying software security based on static analysis alerts and software metrics. *Softw. Qual. J.* **2021**, *29*, 431–507. [CrossRef]
79. Kraemer, H.C. Kappa coefficient. In *Wiley StatsRef: Statistics Reference Online*; Wiley: Hoboken, NJ, USA, 2014; pp. 1–4.
80. Elrhaffari, I.; Roudies, O. Capturing Software Security Practices using CBR: Three Case Studies. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*. [CrossRef]
81. Williams, L.; McGraw, G.; Migues, S. Engineering Security Vulnerability Prevention, Detection, and Response. *IEEE Softw.* **2018**, *35*, 76–80. [CrossRef]
82. Luburić, N.; Sladic, G.; Slivka, J.; Milosavljevic, B. A Framework for Teaching Security Design Analysis Using Case Studies and the Hybrid Flipped Classroom. *ACM Trans. Comput. Educ.* **2019**, *19*, 1–19. [CrossRef]
83. Thai, M.; Sen, A.; Das, A. ACM SIGMETRICS International Workshop on Critical Infrastructure Network Security. *ACM SIGMETRICS Perform. Eval. Rev.* **2019**, *46*, 48–49. [CrossRef]
84. Oveisi, S.; Farsi, M.; Moeini, A. Software Safety Design in requirement analysis phase for a control systems. In Proceedings of the 12th International Conference on Engineering & Technology, Athens, Greece, 28–30 August 2019.
85. Nguyen, J.; Dupuis, M. Closing the Feedback Loop Between UX Design, Software Development, Security Engineering, and Operations. In Proceedings of the SIGITE '19: Proceedings of the 20th Annual SIG Conference on Information Technology Education, Tacoma, WA, USA, 3–5 October 2019. [CrossRef]
86. Aruna, E.; Rama Mohan Reddy, A.; Sunitha, K. Secure SDLC Using Security Patterns 2.0. In *IOT with Smart Systems*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 699–708.
87. Mahendra, N.; Muqeem, M. Validation of the SMBC Framework of Security Testing Using Analytic Hierarchy Process. *ICIC Express Lett. Part B Appl. Int. J. Res. Surv.* **2021**, *12*, 383–393.
88. Mishra, A.D.; Mustafa, K. Security requirements specification: A formal method perspective. In Proceedings of the 2020 7th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 12–14 March 2020; pp. 113–117.
89. Abdellatif, L.; Chhiba, M.; Tabyaoui, A.; Mjihil, O. MDA Approach for Application Security Integration with Automatic Code Generation from Communication Diagram. In Proceedings of the International Conference on Information Technology and Communication Systems, Khouribga, Morocco, 28–29 March 2017; pp. 297–310.
90. Mufti, Y.; Niazi, M.; Alshayeb, M.; Mahmood, S. A readiness model for security requirements engineering. *IEEE Access* **2018**, *6*, 28611–28631. [CrossRef]
91. Kumar, R.; Baz, A.; Alhakami, H.; Alhakami, W.; Baz, M.; Agrawal, A.; Khan, R.A. A hybrid model of hesitant fuzzy decision-making analysis for estimating usable-security of software. *IEEE Access* **2020**, *8*, 72694–72712. [CrossRef]
92. Abdymanapov, S.; Muratbekov, M.; Altynbek, S.; Barlybayev, A. Fuzzy Expert System of Information Security Risk Assessment on the Example of Analysis Learning Management Systems. *IEEE Access* **2021**, *9*, 156556–156565. [CrossRef]
93. Ali, I.; Asif, M.; Shahbaz, M.; Khalid, A.; Rehman, M.; Guergachi, A. Text categorization approach for secure design pattern selection using software requirement specification. *IEEE Access* **2018**, *6*, 73928–73939. [CrossRef]
94. Rafi, S.; Yu, W.; Akbar, M.A.; Alsanad, A.; Gumaei, A. Prioritization based taxonomy of DevOps security challenges using PROMETHEE. *IEEE Access* **2020**, *8*, 105426–105446. [CrossRef]
95. 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]
96. Huang, G.; Li, Y.; Wang, Q.; Ren, J.; Cheng, Y.; Zhao, X. Automatic classification method for software vulnerability based on deep neural network. *IEEE Access* **2019**, *7*, 28291–28298. [CrossRef]
97. Sönmez, F.Ö.; Kiliç, B.G. Reusable Security Requirements Repository Implementation Based on Application/System Components. *IEEE Access* **2021**, *9*, 165966–165988. [CrossRef]
98. Alnabulsi, H.; Islam, R.; Talukder, M. GMSA: Gathering multiple signatures approach to defend against code injection attacks. *IEEE Access* **2018**, *6*, 77829–77840. [CrossRef]

99. Farooq, M.S.; Ahmed, M.; Emran, M. A Survey on Blockchain Acquainted Software Requirements Engineering: Model, Opportunities, Challenges, and Future Directions. *IEEE Access* **2022**, *10*, 48193–48228. [CrossRef]
100. Wang, H.; Zhong, D.; Zhao, T.; Ren, F. Integrating model checking with SysML in complex system safety analysis. *IEEE Access* **2019**, *7*, 16561–16571. [CrossRef]
101. Fusheng, W.; Huanguo, Z.; Mingtao, N.; Jun, W.; Zhaoxu, J. A Novel Key Agreement Protocol Based on RET Gadget Chains for Preventing Reused Code Attacks. *IEEE Access* **2018**, *6*, 70820–70830. [CrossRef]
102. Zhou, Z.; Zhi, Q.; Morisaki, S.; Yamamoto, S. An evaluation of quantitative non-functional requirements assurance using ArchiMate. *IEEE Access* **2020**, *8*, 72395–72410. [CrossRef]
103. Díaz, J.; Pérez, J.E.; Lopez-Peña, M.A.; Mena, G.A.; Yagüe, A. Self-service cybersecurity monitoring as enabler for devsecops. *IEEE Access* **2019**, *7*, 100283–100295. [CrossRef]
104. Hu, Y.; Zhang, Y.; Gu, D. Automatically patching vulnerabilities of binary programs via code transfer from correct versions. *IEEE Access* **2019**, *7*, 28170–28184. [CrossRef]
105. Adhikari, T.M.; Wu, Y. Classifying software vulnerabilities by using the bugs framework. In Proceedings of the 2020 8th International Symposium on Digital Forensics and Security (ISDFS), Beirut, Lebanon, 1–2 June 2020; pp. 1–6.
106. Venson, E.; Alfayez, R.; Gomes, M.M.; Figueiredo, R.M.; Boehm, B. The impact of software security practices on development effort: An initial survey. In Proceedings of the 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Recife, Brazil, 19–20 September 2019; pp. 1–12.
107. Alenezi, M.; Khan, F.I. Context-Sensitive Case-Based Software Security Management System. In *Intelligent Systems Applications in Software Engineering*; Silhavy, R., Silhavy, P., Prokopova, Z., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 135–141.