Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.057587

REVIEW

Review of Techniques for Integrating Security in Software Development Lifecycle

Hassan Saeed[1], Imran Shafi[1], Jamil Ahmad[2], Adnan Ahmed Khan[3], Tahir Khurshaid[4,*] and Imran Ashraf[5,*]

[1]College of Electrical and Mechanical Engineering, National University of Sciences and Technology (NUST), Rawalpindi, 46604, Pakistan

[2]Department of Computer Science, Abasyn University Islamabad Campus, Islamabad, 45510, Pakistan

[3]Military College of Signals, National University of Sciences and Technology (NUST), Rawalpindi, 46600, Pakistan

[4]Department of Electrical Engineering, Yeungnam University, Gyeongsan, 38541, Republic of Korea

[5]Department of Information and Communication Engineering, Yeungnam University, Gyeongsan, 38541, Republic of Korea

*Corresponding Authors: Tahir Khurshaid. Email: tahir@ynu.ac.kr; Imran Ashraf. Email: imranashraf@ynu.ac.kr

Received: 21 August 2024 Accepted: 05 December 2024 Published: 03 January 2025

ABSTRACT

Software-related security aspects are a growing and legitimate concern, especially with 5G data available just at our palms. To conduct research in this field, periodic comparative analysis is needed with the new techniques coming up rapidly. The purpose of this study is to review the recent developments in the field of security integration in the software development lifecycle (SDLC) by analyzing the articles published in the last two decades and to propose a way forward. This review follows Kitchenham's review protocol. The review has been divided into three main stages including planning, execution, and analysis. From the selected 100 articles, it becomes evident that need of a collaborative approach is necessary for addressing critical software security risks (CSSRs) through effective risk management/estimation techniques. Quantifying risks using a numeric scale enables a comprehensive understanding of their severity, facilitating focused resource allocation and mitigation efforts. Through a comprehensive understanding of potential vulnerabilities and proactive mitigation efforts facilitated by protection poker, organizations can prioritize resources effectively to ensure the successful outcome of projects and initiatives in today's dynamic threat landscape. The review reveals that threat analysis and security testing are needed to develop automated tools for the future. Accurate estimation of effort required to prioritize potential security risks is a big challenge in software security. The accuracy of effort estimation can be further improved by exploring new techniques, particularly those involving deep learning. It is also imperative to validate these effort estimation methods to ensure all potential security threats are addressed. Another challenge is selecting the right model for each specific security threat. To achieve a comprehensive evaluation, researchers should use well-known benchmark checklists.

KEYWORDS

Software development lifecycle; systematic literature review; critical software security risks; national institute of standards and technology; DevSecOps; open web application security project; McGraw's touch points

1 Introduction

In the current era of the 5G revolution where mobile phones are easily accessible to all masses, cyber security has gained paramount importance. Cyber-attacks have increased significantly and necessitate increased protective measures. Security measures in application development have gained substantial importance in recent times owing to hacking and other attacks on computer systems. As a result, security has to be made an intrinsic component of all the stages of mobile application development. If sensitive data is hacked or exposed to intruders, it may cause unrepairable losses to software companies' reputations with partners, customers, and investors. Therefore, system managers and users are paying more and more attention to this important aspect of security. To have complete secure applications/software, security aspects must be considered in every step of the software development life cycle (SDLC) or, application development life cycle (ADLC). To integrate security into the software engineering model, it should be considered from the beginning of the SDLC/ADLC [1]. Most organizations normally consider security as a post-development process. Security concerns, if neglected, contain a profound bearing on the overall quality of software, as organizations grappling with insecure software are compelled to address and rectify unreliable applications, while also impeding the progress of other concurrent projects. Emerging cyber vulnerabilities, both internal and external to the organization, continue to surface persistently, posing formidable financial risks and jeopardizing the integrity of critical company data. The repercussions of such security lapses extend beyond monetary losses, encompassing substantial credibility deficits that can be detrimental to the reputation and standing of the organization [2]. In this study, the literature on traditional security integrating models in SDLC has been reviewed to discuss the challenges and limitations of these methods, such as the lack of quantifiable parameters to address each risk involved. Moreover, giving security the utmost priority while developing any software/application is the need of the hour to feel more secure in the cyber world.

1.1 Objectives

The main objectives of this review study are:

- To evaluate various existing techniques for incorporating security in the software development lifecycle and assess risk estimation technique's usefulness in effectively mitigating security risks.
- To find trends, challenges, and probable prospects of integrating risk estimation techniques into software development to make the software more secure.
- To identify research gaps, and future work and give recommendations for developers and researchers on integrating security aspects into the software development lifecycle.

1.2 Research Questions

Based on the objectives, the following research questions (RQs) have been formulated:

RQ-1: How effectively security can be integrated into the SDLC through the use of risk estimation techniques?

RQ-2: Which approaches are effective for software effort estimation?

RQ-3: What are the prospects and challenges in integrating risk estimation techniques into SDLC to increase software security?

The structure of the paper is manifested simply and methodically where literature on existing methods/practices on secure software development framework is presented in Section 2. The proposed methodology is discussed in Section 3. Results are discussed in Section 4. In the end, Section 5 concludes this study and provides future work.

2 Literature Review

Security considerations during software development have been a growing research field, especially in the last decade or so. To study ongoing research trends on integrating security in software, articles published between 2006 and 2024 are reviewed from prominent digital libraries, including Elsevier, IEEE, Web of Science, etc. Various important parameters like inclusion and exclusion criteria based on period, relevance & accessibility are considered for this review.

2.1 Security through Lens of Various Phases of Software Development Lifecycle

Initial Planning and Design. Threat modeling and secure design are the two main cardinals of this phase. Security threats are resolved through threat modeling by devising counter measures for them, while secure design principles call for such application design which ensures enhanced security. A study reveals that most developers do not incorporate security during this phase, and threat modeling is rarely practiced which is why security lapses are observed later on [3].

Implementation. Secure coding strategies/code reviews are the two main activities performed in this phase to ensure that code is functionally correct and resistant to known vulnerabilities. Some developers prioritize secure coding; others overly rely on frameworks without fully understanding their security implications [3].

Testing: Known secure testing methods, such as dynamic and static analysis, are used to reveal and identify susceptibilities that may have gone unnoticed in early phases. It also includes code reviews and penetration testing. Security testing is hardly the focus of developers as the main focus is primarily on functional testing, which again impedes checking security concerns at this stage.

Deployment. To keep the application's security operational. Secure deployment practices and Vulnerability management are key mechanisms that ensure the operation-ability of applications with security mechanisms intact.

Maintenance. Constant monitoring and incident response are vital for maintaining the application's security post-deployment.

'Integrating security' mentions the systematic addition of security measures and concerns at each phase of the SDLC. This scheme is essential for building applications that are resilient to cyber-attacks, protecting users and software producers from any losses and breaches. Integration of security to SDLC is not only a technical requirement but it can also be termed as a key business decision as primarily it serves to reduce weaknesses and post-deployment security fix costs. As a normal practice, historically security was often neglected/ignored throughout the creation of applications. This reactive scheme has the inherent probability of more susceptibilities and exploitation threats. With security-centric methodologies like DevSecOps gaining a reputation for integrating security into every stage of the software development process, secure practices in SDLC have evolved over a short period to become more proactive.

2.2 Strategies and Practices

Coding while keeping security risks in mind helps to end vulnerabilities at the start; this is known as secure coding techniques. Coding standards must be strictly followed to prevent security-related issues, Moreover, code analysis procedures must also be put to use to address security problems [4,5].

2.3 *Security Requirements Engineering*

Security requirements must be properly logged, deliberated upon, and documented in the initial development stage. It ensures that the program is built while keeping security considerations alive during all phases of development [6]. Threat analysis in software development has been highlighted as an effective tool to meet challenges and propose an automated tool for enhancing secure development practices. Automated tools have been introduced to increase the accuracy and effectiveness of threat analysis [7]. The role of security automation throughout SDLC phases to strengthen software security against increasing cyber threats has also been reviewed. It calls for increased security automation from development to deployment [8]. Moreover, it is very critical to add security requirements into traditional SDLC to lessen software vulnerabilities, thereby providing a security requirements tree and verification artifacts for secure design and implementation [9].

In a nutshell, secure software development practices advocate a proactive strategy towards security throughout the lifecycle of software hence highlighting the importance of secure design, coding practices, and adherence to standards for mitigating risks [10].

2.4 *Factors Influencing Security*

Some major security lapses exist in different organizations. Rather, the surprising fact is that there is inequality in how security is adopted in the SDLC since some teams ignore security. The most notable issue is the gap between security practices and their actual adoption, often hampered by the organization's culture and the employees' lack of knowledge on security. In some cases, company sizes also don't correlate with security practices. Another important factor that influences security is the human element, as a considerable number of developers do not have any professional security training and do not consider security as their responsibility [3].

2.5 *DevSecOps*

DevSecOps's main focus is automating security integration into all phases of the DevOps life cycle, from initial design to integration, testing, deployment, and delivery. However, this integration of security involves its own set of challenges [5]. One of the challenges is to make security adaptable to the DevOps processes, which means the security methods need to be highly agile, and the procedures must be accepted and understood by all the teams involved including security, development, and operations teams [11]. Another challenge is how different organizations will start DevSecOps practices by adopting the change in skills, tools, standards, and processes to successfully implement security into their culture [12]. The dynamic nature of the environment implies that critical security functionality should be ready for use in tools that work on the right platforms. Additionally, automation and being conversant with relevant tools and technology is an ongoing challenge for DevSecOps projects.

2.6 *NIST Framework for Secure Software Development*

With the increase in cybercrimes over the last few years, a growing realization of the need for software/cyber security has begun. Unfortunately, being aware that cyber security is something that needs to be worried and knowing what steps to take are two different things entirely. United States', National Institute of Standards and Technology (NIST) developed a framework called cyber security framework (CSF) to help critical organizations in determining what they need to secure their computer systems and networks. Though designed for organizations, much of the guidance provided by the CSF, especially the basic functions is also valuable for communities attempting to put together a community cyber security program [13].

The framework's core functions, as shown in Fig. 1 include 'Govern', 'Identify', 'Protect', 'Detect', 'Respond', and 'Recover' to organize cyber security outcomes at the best possible level.

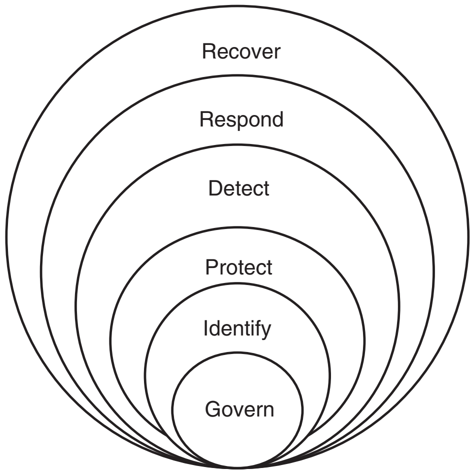

Figure 1: NIST cyber-security framework

Govern. Establish and monitor the organization's cyber security risk management strategy, expectations, and policy. The 'Govern Function' is cross-cutting and provides outcomes to inform how an organization will achieve and prioritize the outcomes of the other five Functions in the context of its mission and stakeholder expectations. Governance activities are critical for incorporating cyber security into an organization's broader enterprise risk management strategy. Govern directs an understanding of organizational context; the establishment of cyber security strategy and cyber security supply chain risk management; roles, responsibilities, and authorities; policies, processes, and procedures; and the oversight of cyber security strategy.

Identify. Help determine the current cyber security risk to the organization. Understanding its assets (e.g., data, hardware, software, systems, facilities, services, people) and the related cybersecurity risks enables an organization to focus and prioritize its efforts in a manner consistent with its risk management strategy and the mission needs identified under Govern. This Function also includes the identification of improvements needed for the organization's policies, processes, procedures, and practices supporting cyber security risk management to inform efforts under all six Functions.

Protect. Use safeguards to prevent or reduce cyber security risk. Once assets and risks are identified and prioritized, protect supports the ability to secure those assets to prevent or lower the likelihood and impact of adverse cyber security events. Outcomes covered by this Function include awareness and training; data security; identity management, authentication, and access control platform security (i.e., securing the hardware, software, and services of physical and virtual platforms), and the resilience of technology infrastructure.

Detect. Detect enables timely discovery and analysis of anomalies, indicators of compromise, and other potentially adverse cyber security events that may indicate that cyber security attacks and incidents are occurring.

Respond. Act against a detected cyber security incident. Respond supports the ability to contain the impact of cyber security incidents. Outcomes within this function cover incident management, analysis, mitigation, reporting, and communication.

Recover. Restore assets and operations that were impacted by a cyber security incident. Recover supports the timely restoration of normal operations to reduce the impact of cyber security incidents and enable appropriate communication during recovery efforts [13].

2.7 Open Web Application Security Project

The open web application security project (OWASP) provides guidelines and essential tools to developers to improve software security by mitigating and identifying vulnerabilities in web applications. Across the literature reviewed, it is evident that OWASP is not only effective in addressing critical security concerns but is also practical in terms of its integration within various secure software development lifecycles (SSDLC) [14]. OWASP Top 10 is a foundation document that addresses and identifies the top ten most critical security risks. It also provides actionable recommendations for software developers to improve their security position [15].

Through comparative analysis, OWASP's comprehensive, lightweight application security process (CLASP) is evaluated against Microsoft's Security Development Lifecycle (SDL) [16]. Such kinds of studies reveal frameworks of both processes, which are aimed to embed security in the development process. However, OWASP CLASP offers more flexibility and is better suited for organizations that are more forthcoming to security and are less rigid [17]. Practical aspects of securing a practical business domain using OWASP guidelines have also been reviewed. Tools like Jenkins and Wireshark etc., which can effectively detect and mitigate security risks such as SQL injection and cross-site scripting (XSS) can be used for this purpose [18]. Literature regarding how the vulnerability is predicted early and how the security risk detection process can be integrated into the SDLC using OWASP guidelines has been reviewed. This proactive idea of identifying risks early in the development phase helps hassle-free working of software in the deployed state [19].

It has also been found, how OWASP guidelines can serve as a foundational framework for small and medium-sized enterprises. The documentation and educational resources surfaced through OWASP's top 10 risks are crucial for these organizations to build and maintain secure software [19,20]. The implementation of SSDL in startup environments has also been reviewed. Indicators that influence successful adoption are identified, indicating how OWASP guidelines can be costumed to meet the unique needs of startups while keeping robust security practices in place [21].

Overall, OWASP provides a critical benchmark for developers to enhance software security through practical guidelines, comparative analyses, and educational resources. OWASP's top ten risks identified are given in Table 1 [15,22].

Table 1: OWASP top 10 risks 2022

OWASP risks	Description
Injection	Injection flaws, such as NoSQL, LDAP injection, etc., occur when untrusted data is sent as part of a query or instruction.
Broken authentication	Broken authentication susceptibilities allow attackers to compromise passwords, and keys and eventually lead to unauthorized access.

(Continued)

Table 1 (continued)

OWASP risks	Description
Sensitive data exposure	Sensitive data exposure occurs when sensitive data is improperly protected hence allowing unauthorized exposure of data to potential attackers.
XML external entities (XXE)	It is when an application processes XML input from untrusted sources.
Broken access control	Broken access control hazards allow attackers to access data without proper authentication/authorization.
Security misconfiguration	Security misconfigurations are due to insecure default configurations, open cloud storage incomplete configurations, etc.
Cross-site scripting (XSS)	XSS susceptibilities allow attackers to inject malevolent scripts into desired web pages.
Insecure deserialization	Insecure deserialization vulnerabilities allow attackers to gain unauthorized access influence serialized objects, and execute arbitrary code.
Using components with known vulnerabilities	If components with known vulnerabilities are used, it exposes applications to attackers.
Insufficient logging & monitoring	Insufficient logging and monitoring impede detection and response to security incidents.

2.8 Microsoft Security Development Lifecycle

Microsoft security development lifecycle (Microsoft SDL) is an innovative approach to software development. It was introduced by Microsoft to address ever-growing security concerns, looming over the software development process. The main focus is on integrating security into the development process from its inception to deployment and beyond [23]. SDL integrates security into every phase of software development and has evolved into an extensive and comprehensive framework [24]. It not only helps in mitigating security risks but fosters a security-aware culture among developers. Microsoft SDL systematically detects, assesses, and mitigates probable security risks throughout the software development process through a regulated, multi-phase methodology [25]. The lifecycle begins with educational awareness, ensuring all team members understand the importance of security.

SDL also adds a few essentially important stages like risk assessment and analysis, where threats are evaluated using methodologies like threat modeling [26] to predict and address software vulnerabilities. Microsoft SDL calls for the development of security documents and tools throughout these critical phases that overall increase the security posture of the software system [27]. The lifecycle concludes with a specifically designed security-centered final review to address emerging threats post-deployment. SDL focuses on rigorous security testing to ensure a higher level of compliance with security standards and regulations, making it suitable for industries where regulatory requirements are strict [26]. SDL methodology has also been applied to various working models to see threat scenarios in factory environments, stretching its use beyond traditional software development situations [28]. So Overall, Microsoft SDL advocates for a proactive, systematic approach to detecting and mitigating security risks throughout the software development lifecycle [29]. By integrating security considerations early, SDL not only enhances software security but also fosters a culture of security awareness and continuous. SDL's role in ensuring robust, secure software remains crucial in conservation against evolving cyber threats in the wake of the ever-increasing role of software applications in our lives.

2.9 MCGRAW's Security Touch Points

Gary McGraw's work in proposing security 'Touch Points' is foundational as it introduces a comprehensive methodology for integrating security into the software development lifecycle. The approach, emphasizes embedding security measures from the beginning of the development phase through to deployment and beyond. This proactive approach helps in mitigating risks that are often neglected in traditional software development models [30]. Table 2 describes McGraw's touch points.

Table 2: McGraw's touch points

Touchpoint	Description
Code reviews	Identification of security vulnerabilities through code review.
Architectural risk analysis	Identification of probable security risks and analysis of the system architecture.
Penetration testing	Attacks are simulated to find security weaknesses.
Security testing	Implementing and executing security-related test cases for validating the effectiveness of security controls.
Abuse cases	Scenarios in which the system could be misused are created and analyzed.
Security requirements	Based on potential security vulnerabilities identified, security requirements are identified.
Security operations	Throughout the system's lifecycle, operational procedures are established to maintain security.

McGraw's methodology stands out due to its comprehensive nature. It identifies and addresses security issues at the beginning of the development process, thereby reducing the chances of the likelihood of vulnerabilities later on [31]. Touchpoints methodology promotes a culture of shared responsibility for security within development teams by nurturing the culture of a shared/cooperative environment among developers and security professionals [30,32]. It also enhances the developer's engagement and awareness towards security, which in turn leads to enhanced and better outcomes in building a protected and secured system. Overall McGraw's touch points because of their adaptability and flexibility to various development models such as Agile, DevOps, etc. provide a robust framework for building secure software [33,34].

2.10 Need for a Holistic Approach based on Research Gaps

Most studies focus on a specific framework or model but do not provide a hybrid and more holistic approach that can address all gaps discussed/identified in each security integrating model. There is a need for empirical studies that compare the effectiveness, accuracy, and scalability of these frameworks to provide developers with clearer guidance on which approach or combination of approaches is most suitable for their specific needs.

3 Methodology

It is imperative to follow a systematic approach in a systematic literature review to minimize biases in the research. This review follows the review protocol suggested by Kitchenham et al. [35] as depicted in the flowchart given in Fig. 2 which divides the review process into three main stages: Planning, Execution, and Analysis. The following steps were implemented for this systematic review.

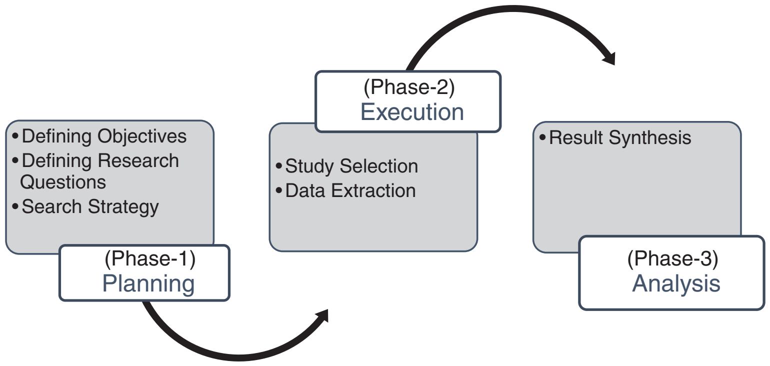

Figure 2: SLR protocol

3.1 *Structural Overview of the Review*

As per Kitchenham's protocol [35], research methodology has been implemented as the second stage to identify related articles based on relevant parameters within the research field.

3.1.1 Planning

Defining Research Objectives

Setting objectives for this review is the first step which aims to achieve effectiveness of security integration into the SDLC through risk estimation techniques, explore various approaches, and identify future challenges.

Defining Research Questions

Based on the research objectives, frame the research questions focusing on how security can be integrated into the SDLC through risk estimation techniques and some other methodologies.

Search Strategy

Relevant literature was identified from various reputable academic databases using some predefined search strings.

3.1.2 Execution

Study Selection

The inclusion and exclusion criteria were employed to study the search results, whereby attention was given to peer-reviewed journal articles and papers that directly discuss security integration in the SDLC and risk estimation techniques.

Data Extraction

Data extraction processes were performed to extract the key variables in the selected studies which include security effectiveness, risk estimation accuracy, and scalability.

3.1.3 Analysis

The results of the selected studies were grouped (Results Synthesis) according to the variables described above. This allowed us to see some of those patterns and trends, as well as gaps in current research.

3.2 Overview of Selected Articles

Initially, for this review, 638 papers were studied. Various search engines were consulted for this review, such as Springer, IEEE Xplore, etc. [36]. These search engines have been selected based on their reputation for hosting high-quality scientific literature in the field of computer software, including software development and cyber security [37]. Some of the major keywords that were used in the search string(s) for searching from databases are:

- Secure Software Development Lifecycle.
- Risk Estimation Techniques.
- DevSecOps.
- OWASP security guidelines.
- Natural Language Processing for vulnerability detection.
- Security Assurance Model.
- McGraw's Touch Points.

The details of the paper selected from each repository are shown in Fig. 3.

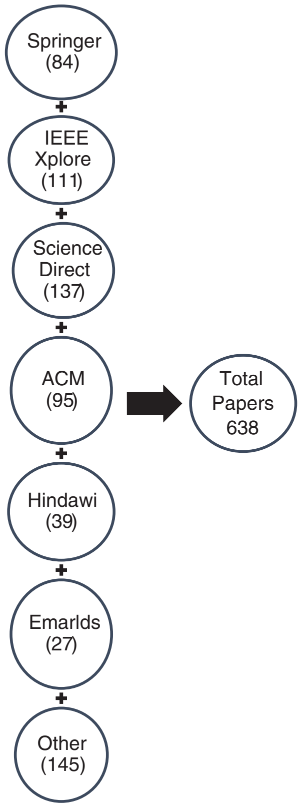

Figure 3: Papers reviewed from various searches

Detail of the 638 papers reviewed includes publication year, number of citations, etc., paper studied initially were from the year 1999 to 2024. The number of citations is another metric for quality evaluation. It also shows the current direction of research. Both are explained in Fig. 4 and Table 3, respectively.

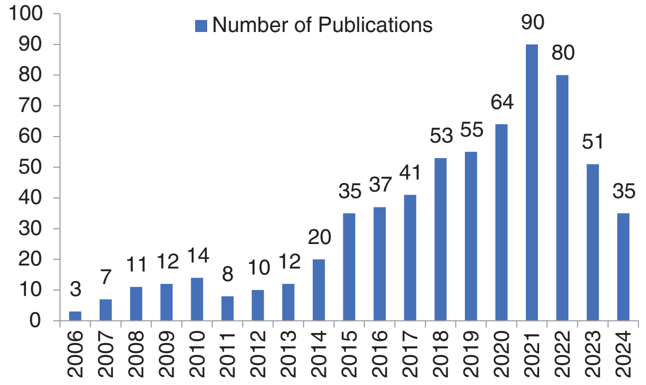

Figure 4: Total research papers by publication year

Table 3: Research papers with citation count

Range	Entries
1 to 10	171
11 to 20	94
21 to 40	89
41 to 75	56
Greater than 75	87
No citation	141
Total	638

In the first stage, only recent papers (not older than the last two decades) were included to ensure the inclusion of modern-day and recent research, narrowing the paper count to 397 papers (Fig. 5).

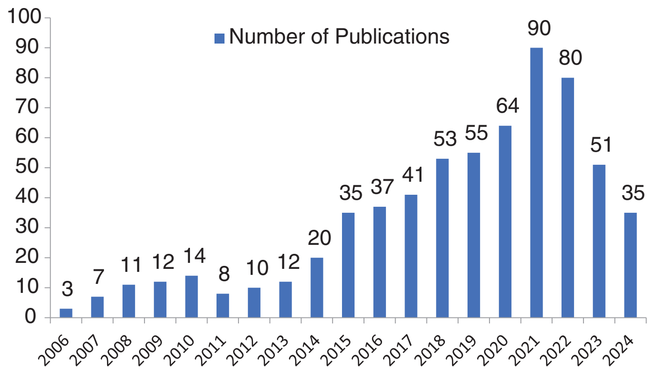

Figure 5: Detail of research papers selected by publication year

In the second stage, the relevance of each paper to integrating security into the SDLC and the impact of effort estimation methods on improving software security using concepts like OWASP, DevSecOps, and protection Poker was assessed, further narrowing the selection to 218 papers.

In the third stage, the quality of the remaining papers was assessed. Only studies that had a clear empirical impact or papers whose comprehensive reviews were present were selected. This step ensured that the selected papers had strong methodologies and significant contributions to the field. Papers that lacked methodological flow or empirical evidence were excluded, further narrowing the selection to 147 papers.

Finally, the fourth level used citation count as a threshold to judge whether the papers were impactful and relevant in the research community. Papers cited by more papers were considered the best probably to be authenticated. This resulted in a final selection of 104 papers, which provided a solid foundation for the study. The complete selection process is depicted in Fig. 6.

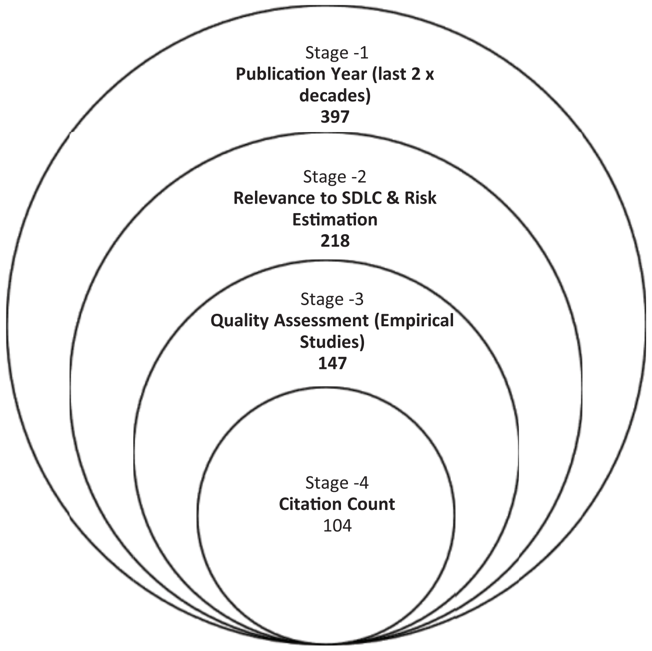
Stage -1
Publication Year (last 2 x decades)
397
Stage -2
Relevance to SDLC & Risk Estimation
218
Stage -3
Quality Assessment (Empirical Studies)
147
Stage -4
Citation Count
104

Figure 6: Stages-wise selection/filtration process

Furthermore, several other factors considered as inclusion and exclusion criteria have been applied to ensure that the articles selected for review meet specific quality and relevance standards.

3.3 Inclusion Criteria

The inclusion criteria are based on various factors [30] for selecting the articles.

Language. Only articles published in the English language are included in the review.

Relevance. The selection criteria for the articles we have included is that they directly address some of the key research themes introduced in Section 2. These themes include the integration of security into the Software Development Lifecycle (SDLC) and techniques to estimate risk while designing or developing, secure software practices. Articles that propose relevant frameworks and contribute to the understanding of these themes are prioritized [38].

Accessibility. Articles easily available and in full-text form are generally included.

Quality. Articles published in well-reputed journals and conferences were given preference for selection.

3.4 Exclusion Criteria

The exclusion criteria used in this review paper [39] are as follows.

Research not addressing security aspects of the SDLC, such as those focusing solely on software performance or usability without security considerations.

Articles not presenting adequate and robust methodological frameworks.

Articles not fully accessible in any capacity.

Non-peer-reviewed articles are not considered for this study.

3.5 Analysis of Selected Variables

We used the following key variables to analyze selected articles. The results section will rely on these variables for structuring the analysis.

Effectiveness. When we talk about the effectiveness of security integration, it means how well we have integrated security practices into each stage of SDLC.

Risk Estimation Accuracy. How many methods developers can use to determine security risks that will be accurate and reliable?

Scalability and Adaptability. This includes how scalable each approach/method is for different sizes and complexity of projects, as well as how adaptable they are to the organization's needs.

3.6 Data Managing and Referencing

In data sifting all the data is organized as per the libraries and it is ensured that all articles are complete. Data is preprocessed through Mendely for further organizing as per requirements. It is a popular tool among academics and scholars because it makes managing research materials simple and efficient. Users can import articles directly from databases, websites, and other sources Mendely is a free and open-source tool Integra table with MS World and allows researchers to save and organize articles, annotate them, and generate bibliographies [40,41].

3.7 Execution of Proposed Methodology

The methodology employed in this research involves the review and analysis of various risk estimation and software vulnerability detection methodologies that can practically mitigate such security vulnerabilities. Almost all these methodologies involve an organized and efficient strategy to assess the likelihood and consequences of security risks, leading to an in-depth understanding of potential threats [42]. It generally involves quantifying a risk level associated with every identified security concern. The following stages are generally extracted from various risk/effort estimation techniques.

Risk Estimation Technique. To effectively evaluate the risks linked to a specific circumstance, it is imperative to measure each risk using some risk estimation technique. Many methodologies are employed to effectively identify and assess risk involved in software development and deployment. Usually, a numerical figure is also linked to these risks identified as it helps in the evaluation of risks and then eventually their impact [43].

Data Analysis. To assess risk levels in a company, it is crucial to compare them against industrial/recommended practices. This comparison enables software houses/firms to identify areas that need additional stringent measures and also in the formulation of efficient risk mitigation plans [44,45]. The process involves pinpointing risks with the highest scores. These risks represent the most significant threats to the organization and demand urgent action. By prioritization of these risks, companies can optimize their resource allocation and focus on resolving the most critical concerns [46].

Risk estimation techniques are the foundation block of many software systems under development as they provide a clear insight about resources, and time required [47,48]. Moreover, it serves as a correct yardstick to practically consider the security risk occurrence probability and due effort required [49].

4 Results and Discussions

The results of this study are classified as key elements of security integration in the SDL [49]. Different methods to ensure security are reviewed in this section and key aspects of how well they handle the security risks during the development process have been analyzed. This is followed by the information gathered from empirical data and literature which emphasizes the necessity of incorporating security controls in different stages of SDLC [50]. Moreover, identifying and using an effective effort estimation method as a risk assessment tool also forms part of this section.

RQ-1: How effectively security can be integrated into the SDLC through the use of risk estimation techniques?

To answer this question, several approaches have been studied to analyze their effectiveness for integrating security into the SDLC.

4.1 Factors Affecting Secure Software Development

Implementation of secure software development practices is very important for the preservation of information and ensuring the integrity of software systems. A Delphi study conducted within Malaysian Public Service Organizations highlights the factors that influence the implementation of secure software development practices [51]. Key factors identified in the study such as organizational commitment, cost, skill level of personnel, and time constraints are significant barriers to the implementation of these practices [52]. Despite the availability of various models and standards for secure software development, practical adoption remains a challenge due to the associated costs and required expertise [53]. The research also indicates the need for a bespoke approach that can consider the specific constraints and requirements of public service organizations to efficiently integrate security practices throughout the software development lifecycle [54].

4.2 Embedding Security Measure into Software Development Lifecycle

Researchers have also proposed the idea of integrating security measures into the software development lifecycle from the start by instituting application confidentiality, integrity, availability, non-repudiation, and authentication (CIANA) into software development. This idea surfaces due to the realization that the software development lifecycle fails to fully integrate security practices, as they only combine security training and incident response processes with traditional development steps. So, it calls for a more holistic approach where security considerations are embedded throughout the development process, starting from the initial planning and design phase. The proactive approach which calls for the need for developing models, tools, architectures, and algorithms that support

CIANA principles from the start of software development can help mitigate security risks and ensure that applications are robust towards cyber threats [55].

4.3 Natural Language Processing-Machine Learning Ensemble for Automatic Detection of Security Vulnerabilities

The literature explores the integration of software security and natural language processing (NLP) techniques, focusing on the growing need to tackle software security. Conventional methods like code analysis are cumbersome and have proven to be less effective. Hence researchers reviewed and proposed various models and approaches to automate the security vulnerabilities detection process in software systems. In such techniques, source code is treated as text for NLP and ML techniques.

Various researchers have modeled systems that can automatically detect software vulnerabilities. The accuracy varies from 88% to 95% depending from model to model. One such model uses deep learning NLP techniques; its accuracy is around 93%. It performs preprocessing of the data through NVD/SARD databases (which leads to a comprehensive database C programming files) carrying only known vulnerabilities [56]. The prospect of the use of NLP and deep learning models to cover the issues of late-stage security incorporation in software development has also been explored [57]. It is often effective to quantify security effort in the software development lifecycle through the word (topic) selection on text artifacts from version control systems (VCS), issue tracking systems, etc. [58]. Similarly, another novel technique has also surfaced where the Word2vec model and random forest classifier are used to normalize and extract features from software functions, hence leading to the effective detection of security vulnerabilities [59]. Similarly, comparison-based systems have also been proposed where deep learning models are developed for software vulnerability detection, attaining a high accuracy of about 95% accuracy [60]. Doc2vec NLP algorithm has also been in use for making sentence vectors from system logs by applying classification algorithms to detect software vulnerabilities [61,62].

Collective findings from all the literature studied on NLP-ML amalgamation, highlight the efficacy of NLP and ML in automating the detection of software vulnerabilities. The models proposed in these papers demonstrate high accuracy and efficiency, hence overcoming the limitations of traditional code analysis methods [50,63].

4.4 Prominent Risk Estimation Models

There are various risk estimation models in the field of software development like FMEA, STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege), DREAD (Damage, Reproducibility, Exploitability, Affected Users Discoverability), Protection Poker, OCTAVE (Operationally Critical Threat Asset and Vulnerability Evaluation), PRA (Probabilistic Risk Assessment) and TARA (Threat Agent Risk Assessment) [64]. All methods have their part to play in risk assessment, but only a few are mainly designed for use during the software development phase. Since this research is solely focused on the development process FMEA, STRIDE, DREAD, and Protection Poker being the models effective and applicable during the entire development process and not after deployment are discussed only.

4.4.1 Protection Poker

It is a collaborative and lightweight technique designed to help agile teams estimate security risks. Various benefits are associated with Protection Poker, including improved discussions on security, increased knowledge and awareness among team members, and contributions to defining security requirements. However, there are certain challenges as well, notable challenges include the time required to play the game, confidentiality in the results, and integrating outcomes in the development process [65]. Adaptation of the Protection Poker Technique can help agile developers to effectively incorporate security risk assessment methods into their practices, enhancing the overall security of their software products.

The use of Protection Poker as a practical tool is explored for software security prospects in agile development. Protection Poker is a collaborative technique that helps software teams identify and assess security risks early in the development process. Protection Poker is an effective tool for agile teams if properly integrated into their work methodology [66,67]. In an experiment conducted with students working in Scrum teams at the University of Oslo, Protection Poker was used to identify and address security and privacy risks during the software development process. The experiment showed how it fosters team discussion and convergence of opinion in an iterative manner, hence resulting in compliance with regulations and building secure systems in a collaborative manner [68].

4.4.2 Fault Mode Effect Analysis (FMEA)

It is a proactive approach that is used during the software development phase to find potential failure modes in the system and evaluate their causes and impact on the project. FMEA is particularly effective in preventing problems before occurring by looking at potential failures that exist within the software development phase [69]. This analysis is carried out in six steps, which are given in Table 4.

Table 4: FMEA's risk estimation process

Number	Step	Description
Step 1	Identify failure modes	Identify probable failures and to analyze what could go wrong in the system.
Step 2	Assess severity	In this step, if failure occurs severity of the impact is evaluated.
Step 3	Determine occurrence	Estimate the likelihood of the failure.
Step 4	Assess detectability	Before the failure can cause any trouble, evaluate how easily it can be detected.
Step 5	Calculate risk priority number (RPN)	Calculation formula: Severity × Occurrence × Detection.
Step 6	Prioritize and mitigate	Use the RPN to prioritize, higher RPN number risks be addressed first as they are more critical.

4.4.3 STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege)

It is a threat modeling technique used to recognize the potential threats occurring in phases of software development that are written with only security in mind. This model divides threats into 6 types so developers may understand where their system might be at risk and what sort of stuff comes under certain security issues [70]. STRIDE helps us to decide the most appropriate development strategies that contribute towards rectification of these threats and hence maintaining secured software before its deployment [71]. Details of six security threats are given in Table 5.

Table 5: STRIDE's threat model

Category	Description
Spoofing	Mimicking another process or user (e.g., fake identity or credentials).
Tampering	Unlawful amendment in data or system components.
Repudiation	Denial that an action or operation took place without proper auditing.
Information	Disclosure Exposure of sensitive or confidential information to unauthorized parties.
Denial of service	Unavailability of services or systems, usually through resource exhaustion.
Elevation of privilege	Gaining unauthorized control of prohibited data.

4.4.4 DREAD

It is a risk estimation model that helps in handling security vulnerabilities based on five factors: Damage, Reproducibility, Exploitability, Affected Users, and Discoverability. The tool will give a numeric score to each of the factors which enables developers to work on what they should pay more attention to from a security point of view. During security reviews in the software development process, it is used as a risk assessment model to help identify and prioritize risks that could influence the security of the system [72]. This way development teams can work on a thorough list of fixes starting with critical ones. DREAD's factors are given in Table 6.

Table 6: DREAD's five factors

Factors	Description
Damage	Degree of damage or harm caused by the threat.
Reproducibility	Ease with which attack or vulnerability can be reproduced.
Exploitability	Ease with which vulnerabilities can be exploited by attackers.
Affected users	Number of users or systems affected by the vulnerability.
Discoverability	The ease with which vulnerability can be discovered by the attackers.

These risk estimation techniques cater to different levels of specificity that may be needed in software development. FMEA is a well-organized, top-down approach that enables the robust detection of risks, STRIDE is particularly useful for a rough categorical breakdown of threats, and DREAD is generally a more straightforward risk prioritization method. Protection Poker on the other hand emphasizes collaboration and team input, making it ideal for agile environments. Four simple strategies that have clear strengths and weaknesses are compared with their practical application mentioned in Table 7.

Table 7: Comparison of risk estimation methods

Method	Practical application	Strengths	Weaknesses
FMEA	Classifying different modes of failures in software design	Systematic, quantifiable, early risk detection	Time-consuming, do have a rare miss rate of critical risks
STRIDE	Identifying security threats in software systems	Prime focus on security threats	Functional or operational risks are not covered
DREAD	Prioritizing security vulnerabilities	Simple, numeric scoring for prioritization	Subjective scoring only, mainly used in the design and architecture phase
Protection poker	Collaborative risk estimation in agile teams	Encourages team collaboration, easily Integra table in agile model	Less formal, team members may lack depth in technical risk evaluation

4.5 Risk Estimation Methodologies in Real-World Applications

In dynamic environments such as DevOps and Agile, the speed at which development is carried out makes its own set of challenges on how risk assessment methodologies like FMEA, STRIDE, DREAD, or Protection Poker can be used. Continuous integration/Continuous deployment (CI/CD) is the main cardinal of dynamic environments like DevOps, where any changes to the code are merged into the production system, this creates the need for easily Integra table security techniques that can keep pace with rapid iterations of development.

For instance, methodologies like STRIDE which require structured and detailed analysis of system components may not fit particularly well with the pace required for the iterative nature of dynamic operations, like Agile and DevOps [73]. Similarly, DREAD which requires manual input from team members for scoring also becomes cumbersome [72]. In contrast, the Protection Poker technique may be best suited for dynamic environments as it focuses on teamwork and can be effective in developing a team-wide awareness of security risks [65]. However, protection poker with its reliance on team discussions may slow down the large-scale DevOps pipelines but small-scale DevOps operations may not be affected by Protection Poker [67]. The use of automated tools like OWASP ZAP (application security testing tool) integrated with these traditional methods may enhance their effectiveness in dynamic environments [17].

4.6 Adoption Challenges in Organizations

STRIDE and FMEA can be effectively used for large organizations because of the availability of specialized security teams that have the resources and expertise. On the other hand, smaller organizations/startups or those following Agile methodologies might find it hard to adopt such hefty processes [69,73]. Small teams often do not have the security experts, needed for extensive threat modeling (under STRIDE and FMEA) can take a long time to complete which may not go well with Agile's need for rapid iteration and flexibility. Furthermore, security-focused methods like DevSecOps may face cultural barriers where the development and operations team are required to do more security work in addition to their existing manifesto [11].

Change in the mindset of the team members is one of the significant challenges for smaller organizations where security may not be a top priority because of a lack of resources [5]. Integration of risk estimation techniques is particularly useful for dynamic operations like DevOps where it can be integrated into automated pipelines. It ensures security remains part of the entire development workflow without overburdening developers.

4.7 Analysis Vis-a-Vis Effectiveness, Accuracy and Scalability

Now in this section, all operational and existing approaches (discussed in Section 2) have been evaluated against the key variables of effectiveness in security integration, risk estimation accuracy, and scalability.

DevSecOps. In DevSecOps, security is continuously added to the development process, hence it is flexible but needs a higher degree of interaction between different components of software development. DevSecOps, automated risk estimation tools get more of a black box proposition which provides accuracy depending on the tools used. DevSecOps is highly scalable and suitable for both small and large projects [11].

NIST. It is best suited for incorporating security needs into the work paradigm of various organizations. It inherently caters requirements of large organizations but will have some difficulty meeting specific needs because of its generality [13]. When it comes to the accuracy of NIST, it has better outcomes for larger systems but overall guidance can lose some precision in a quickly changing environment. Scalability aspects are higher in larger systems but are more challenging for smaller systems.

McGraw's Touch Points. Integrates security practices (e.g., code reviews, etc.) into the development cycle, enabling more proactivity, but requires more interactive work ethic from all teams involved. These touch points are adaptable to various development models but may lead to inconsistent applications across projects [30].

Secure SDLC. To have early risk identification and mitigation, security has to be integrated into each phase of the software development lifecycle (SDLC), but this can add complexity and can slow down the speed of development. A systematic measurement approach will provide the greatest level of risk estimation accuracy, but results will vary as per the tool selected. Owing to its rigid structure, it is less adaptable in dynamic environments.

OWASP. It provides security hardening guidelines that are generally accepted and can serve as a template to be applied readily to established systems with little or no modifications. The problem is that OWASP does not cater to all kinds of security risks, particularly new or changing risks. OWASP guidelines are highly scalable given it remains applicable to the system [16,19].

NLP. Works great when it comes to automate security integration. If better platforms or systems are available NLP performs security integration with a fair degree of success. Scalability remains limited due to computational resources and training data [50,57].

Table 8 points out the pros and cons of security frameworks currently used for SDLC security.

Table 8: Pros and cons of various software security integrating models

Framework/Models	Pros	Cons
Secure SDLC	• Integrating security into each phase of SDLC ensures vulnerabilities are identified and mitigated early, reducing the likelihood of security breaches. • Cost-effective by early identification and fixing of security risks. • Regulatory and compliance requirements related to software security are met by adhering to secure SDLC methods.	• Complexity is added to the development process while implementing Secure SDLC. • The process is resource-intensive and demands additional resources in the shape of time, personnel, and financial investment. • Additional focus on security can delay the development process. • It does not sufficiently satisfy the requirements of volatile and changing requirements.
DevSecOps	• Continuous Security Integration. • Enhanced Collaboration between different teams, i.e., design, coding, testing, etc., yields better performance. • DevSecOps practices are highly scalable as per the software requirements demands.	• Diversity is required in skillset among team members. • To adapt DevSecOps fully, organizations may require cultural change, which may lead to resistance within some organizations.
NIST	• Consistency and reliability in security practices are achieved through well-composed standards and guidelines. • The credibility of NIST guidelines is immense and are highly respected/trusted in the industry. • Highly adaptable and can be used for various organizational needs and sizes, providing huge flexibility.	• The wide-range nature of NIST guidelines can be overwhelming and challenging to implement fully. • NIST guidelines are prone to overfitting, especially for ML models within the NIST framework. • NIST framework can be highly inaccurate with noisy data, hence performance is degraded exponentially.

(Continued)

Table 8 (continued)

Framework/Models	Pros	Cons
OWASP	• It can be incorporated into existing systems and hence highly flexible and adaptable. • Due to the availability of open-source tools and resources, its cost is generally considered to be on the lower side. • Helps meet various regulatory and compliance requirements by addressing critical security vulnerabilities.	• Significant initial training may be required before effectively integrating and implementing OWASP guidelines. • Not all potential and probable risks are covered.
Microsoft SDL	• A comprehensive approach integrating security throughout the software development lifecycle. • Proactive identification and mitigation of security vulnerabilities. • Structured phases and rigorous security testing policies ensure high-quality, secure software. • Effective in reducing post-release security issues and vulnerabilities.	• Challenges in scaling to large, complex projects without proper planning. • Continuous updates and adjustments are needed to keep pace with evolving threats. • Dependency on internal Microsoft resources and guidelines for optimal implementation.
McGraw's touch point	• Provides a proactive approach by embedding security from the initial stages of development. • Includes detailed coverage of security incorporating practices, e.g., code reviews, penetration testing, etc. • Flexible and adaptable- can be integrated with various development models like DevOps and Agile, etc.	• Challenging for teams as it requires a change in mindset to support a collaborative approach. • It is resource-intensive. • Because of a less rigid structure, it may lead to different applications with different kinds of projects.

RQ-2: Which approaches are effective for software effort estimation?

4.8 AI and Deep Learning Techniques in Software Effort Estimation

AI along with deep learning techniques have been observed to be quite useful not only in the sector of software risk assessment but by providing the best tools for the prediction, mitigation, and management of risks throughout the SDLC as well. More exactly, there are deep-learning methods like Convolutional Neural Networks (CNNs), etc., and some other machine learning algorithms that have been engaged with the field of software risk prediction early in the software development phase.

Integration of Security Risk Estimation with Artificial Intelligence. Threat detection techniques when integrated with AI techniques can significantly increase by approximately 20% allowing for faster detection of phishing attempts and unauthorized access. Response times can also be significantly reduced by up to 30% using risk assessment methods integrated with AI. This reduction in response time aspect emphasizes the practicality of using AI techniques for real-time risk assessment [74]. There are numerous deep learning (DL) techniques like CNN and Recurrent Neural Networks (RNN) that have utility in cyber security. For security vulnerabilities like Advanced Persistent Threats (APT), ransom ware, and malware intrusions, CNN-based models, achieved detection accuracy rates of up to 98%. However, there are also certain challenges related to the interpretability of these deep learning models, which remains an impediment in deploying these systems for real-time risk assessments [75]. Cyber components such as computers and networks integrated with physical entities like sensors and actuators form cyber-physical systems (CPS) which are used for monitoring and controlling physical systems. A significant improvement of around 50% has been observed in risk prediction accuracy using AI-enabled security risk estimation techniques in cyber-physical systems [76].

Real World Application of AI and NLP for Risk Mitigation. Empirical research shows that models like BERT and XGBoost are particularly useful for threat detection and are more proactive in threat modeling, these models work on analyzing Common Vulnerabilities and Exposures (CVE) reports and real-time cyber security news. This approach automatically detects and assesses risks derived from threats extracted from Natural language texts. The study is greatly beneficial for healthcare environments [77]. Similarly, the application of AI techniques for detecting cyber threats has also been found useful. Through AI when coupled with a vulnerability management system (VMS), organizations were able to reduce vulnerabilities by up to 30% using early detection of threats. This aspect shows the real-world effectiveness of AI tools in improving the accuracy of threat detection and hence reducing false negative and false positive instances by 25%. This is the empirical proof of AI effectiveness in risk mitigation [78].

4.9 Use of Artificial Neural Networks for Effort Estimation

Accuracy in software effort estimation is another critical feature of successful software project management. The use of fully connected artificial neural networks (ANNs) optimized with soft computing techniques for software effort estimation has been evaluated [79]. The research established that the integration of ANNs could greatly improve the accuracy of effort estimation models. The performance of these models against traditional estimation techniques was compared and established that the ANN models provided more reliable and accurate estimates. The study also revealed that ANNs were further refined using soft computing techniques, such as genetic algorithms and fuzzy logic, to further elevate the performance of neural network models hence making them more robust and adaptable to various project scenarios [80]. The research underscores the importance of adopting these advanced estimation models could lead to better resource allocation, reduced project risks, and improved project outcomes [81].

4.10 Application of Machine Learning Techniques for Effort Estimation

A review of machine learning techniques for software effort estimation covers a wide range of machine learning algorithms, including support vector machines (SVMs), neural networks (NN), and other collaborative methods [79]. Significant advantages have been offered by machine learning techniques in terms of accuracy and adaptability, making them all suitable for complex and dynamic software projects [50]. In the case of software effort estimation, there are several challenges associated with machine learning techniques as well. It includes complexity for the high-end and a large system, complexity also exists in the case of model training and validation [82].

The most appropriate machine learning technique for software effort estimation can be selected using a Multi-Criteria Decision Making (MCDM) approach. This selection will enable project managers to avoid budget and time overruns, also it will make it possible to give due diligence to security concerns where due. Many machine learning techniques like Random Forest, Support Vector Regression, etc. have been found to show promising results in effort estimation [72]. Accuracy and reliability of effort estimation are also very critical to ascertain the true potential of various software bugs, potential vulnerabilities that can later become a security concern for the organizations using the software system [83].

4.11 Long Short-Term Memory for Effort Estimation

Application of Long Short-Term Memory (LSTM) networks for software development effort estimation is another prospect to look for LSTM networks, well suited for time series prediction/sequencing is a type of recurrent neural network [84]. A study demonstrated that LSTM networks outshines traditional estimation models, in achieving high accuracy and reliability [85]. The researchers also evaluated the LSTM network's performance with other machine learning models using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and R-squared. The robustness and adaptability of LSTM is an important highlight as it provides better estimates across all kinds of datasets [86].

As software threats continue to grow, they become a major aspect of software quality reflected in the increasing number of discovered and predicted code bugs. To localize software vulnerabilities, analytics plays a key role. It is imperative to carry out security analysis requirements to fix and repair various known bugs and functions. A method has also been proposed by researchers to combine multi-head attention (MHA) and long short-term memory (LSTM) to handle variable-length input sequences, a crucial ability for applications where input length is variable [87,88].

4.12 Utility of AI and Deep Learning in Software Risk Assessment

Risk Prediction through AI and deep learning technologies. It enables project managers to forecast risks and detect them by analyzing project data to uncover patterns indicating potential issues before they escalate. For example, classifiers such, as Naive Bayes (NB), Random Forest (RF), and Decision Trees (DT) have proven to be effective, in identifying risks associated with software requirements [89].

Minimizing Personal Bias. Conventional risk evaluations frequently face challenges due, to the personal biases of evaluators. Application of deep learning methods like LSTM networks can help alleviate these biases by automating the process of risk assessment which results in enhancing the accuracy and reliability of predicting risks. According to a research study conducted on this subject, the adoption of LSTM networks resulted in a 96.92% increase, in the accuracy of risk estimation thereby reducing discrepancies arising from inclinations [90].

Handling Complex Data. When it comes to dealing with large data sets, like descriptions of software requirements and risks that are unstructured and complex, AI methods truly help. For instance, there have been advancements in creating risk prediction models through the utilization of datasets containing a variety of attributes such, as requirement categories and risk probabilities aimed at managing this data and furnishing risk evaluations [89,90].

4.13 Protection Poker an Effort Estimation Method

Protection Poker provides project managers with the leverage to use of collective expertise of their teams, leading to more accurate and reliable estimates. This technique adopts a collaborative project environment. Furthermore, various models and approaches including algorithmic models, expert judgment, and machine learning techniques have been reviewed and assessed. It has been revealed that despite the availability of numerous estimation models, achieving high accuracy remains a significant challenge. Risks associated with underestimation and overestimation have been highlighted, which can lead to project delays, cost overruns, and potential project failure [55].

Protection poker is a beneficial tool as it nurtures security-focused discussions in agile development. It encourages deliberation of threats in real-time, hence assisting teams to align with solutions eventually protecting the system from security breaches. However, the use of protection poker does call for efficient use of resources like time, etc., and ensures an overall positive tangible impact [68].

4.14 Effort Estimation Vis-a-Vis Agile Methodologies

Effort estimation techniques are of utmost importance in agile software development. A comparative analysis of popular software effort estimation reveals the efficacy of the ensemble approach using planning poker and Expert Judgment as the two most effective techniques for effort estimation [82]. To evaluate the efficiency of these techniques, the researchers employed a controlled experiment comprising two groups of professionals. One group used Expert Judgment, while the other used Planning Poker to estimate the effort required for real-life projects. Results revealed that Planning Poker provided more accurate estimates compared to Expert Judgment, highlighting its efficiency and suitability for agile environments [91].

4.15 Planning Poker an Effort Estimation Method

Planning Poker provides project managers with the leverage to use of collective expertise of their teams, leading to more accurate and reliable estimates [92]. This technique also encourages team engagement and adopts a collaborative project environment. Furthermore, various models and approaches, including algorithmic models, expert judgment, and machine learning techniques have been reviewed and assessed. It has been revealed that despite the availability of numerous estimation models, achieving high accuracy remains a significant challenge [81,93]. Risks associated with underestimation and overestimation have been highlighted, which can lead to project delays, cost overruns, and potential project failure [94].

Empirical analysis of different machine learning algorithms for effort estimation revealed the performance of several algorithms using multiple datasets, such as China, Albrecht, and Maxwell. The findings reveal that integrating machine learning approaches can scale up the accuracy of effort estimates, especially fruitful in the planning and execution of software development [95].

4.16 Comparison of Wideband Delphi with Planning Poker

Comparison of planning/protection Poker with wideband Delphi reveals that practitioners and professionals who used these techniques estimated the development effort for web and mobile applications. The comparison results showed that planning poker is more efficient in terms of time and resources than Wideband Delphi [96]. Analysis of Wideband Delhi with planning poker is vital for software project managers who must balance accuracy with efficiency in their estimation processes. Wideband Delphi is preferable for projects where accuracy is critical, whereas Planning Poker is more suitable for projects with tight timelines and limited resources [97]. Project managers can select the most appropriate estimation technique by considering the trade-offs between effectiveness and accuracy for their specific project needs [98].

4.17 Application of Machine Learning Techniques for Effort Estimation

A review of machine learning techniques for software effort estimation models covers a wide range of machine learning algorithms, including support vector machines (SVMs), neural networks (NN), and other collaborative methods [99]. Significant advantages have been offered by machine learning techniques in terms of accuracy and adaptability, making them all suitable for complex and dynamic software projects [43]. In the case of software effort estimation, there are several challenges associated with machine learning techniques as well. It includes complexity for the high-end and a large system, complexity also exists in the case of model training and validation [74].

The most appropriate machine learning technique for software effort estimation can be selected using a multi-criteria decision-making (MCDM) approach. This selection will enable project managers to avoid budget and time overruns, also it will make it possible to give due diligence to security concerns where due. Many machine learning techniques like the random forest, support vector regression, etc. have been found to show promising results in effort estimation [72,83]. Accuracy and reliability of effort estimation are also very critical to ascertain the true potential of various software bugs, potential vulnerabilities that can later become a security concern for the organizations using the software system [100].

RQ-3: What are the prospects and challenges in integrating risk estimation techniques into SDLC with a view to increasing software security?

4.18 Security Integration Methods

A study of 104 articles has manifested that a collaborative approach is necessary to resolve software security risks effectively. The use of a numerical scale for quantifying risks enables a thorough understanding of their severity, permitting dedicated resource allocation which eventually helps in resolving security risks. Analysis of various articles reviewed has stressed some essential techniques like Thread Modeling, Secure Coding Practices, and Security Testing as critical for integrating security into SDLC. Details of the phases of SDLC where these techniques can be integrated are as follows:

Planning and Design Phase. Threat modeling and secure design are vital in the planning and design stage. Threat modeling identifies security risks and formulates corrective measures. The principles of secure design make sure that the application architecture is sturdy against security threats. Studies [7,32] show that threat modeling can effectively mitigate around 50% of hazards. These techniques substantially decrease the risk of undetected trespassing.

Implementation Phase. Code reviews and secure coding practices are imperative in the implementation phase. To eliminate security hazards right from the outset secure coding strategies, play a vital role, Moreover, code reviews are instrumental in ensuring that code remains resilient to security threats. Studies have revealed that integrating security practices early in the development process leads to more secure products with secure coding showing the effectiveness of around 60% in improving overall security [32]. The idea of integrating security hazard mitigating methods into the implementation phase is found to be fruitful [22].

Testing Phase. Generally, the testing phase of software development is the hallmark for ascertaining the quality of the software product. Static and Dynamic analysis methods are used in the testing phase for the detection of any missed security hazards. The role of penetration testing in the identification of security hazards is immense. The effectiveness of Static and Dynamic Analysis in mitigating security risks is found to be around 75% and 80%, respectively, while that of penetration testing is around 70% [7,32].

Deployment and Maintenance. To upkeep the application's security after deployment, it is imperative to use methods like continuous monitoring and vulnerability management. To address any emerging threat, it is generally useful to go for a collaborative approach where immediate incident response is coupled with constant maintenance [7]. These post-deployment methods like vulnerability management are vital for software security in case of long-term security maintenance with its effectiveness found to be around 65% [32]. Table 9 shows the strengths and weaknesses of various security integration models.

Table 9: Strengths and limitations of research articles

Ref.	Approach/Area covered	Strengths	Weaknesses
[4]	Security risks in software development through the application of Security Assurance Model (SAM)	• Comprehensive analysis of security risks posed to software development. • Provides a comprehensive guide for general readers.	• Lacks comparative analysis of the techniques studied. • Long tabulated data based on RQs can be monotonous.
[21,22]	OWASP	• Provides guidelines and essential tools to improve software security. • Mitigates and identify vulnerabilities in web applications.	• Significant initial training. • Limited risk coverage.
[30]	McGraw touch points	• Results are supported with figures, increasing comprehension for readers. • Implemented and verified on real working software.	• No use of automated tools for collecting security measurement data. • Resource Intensive Implementation.
[62]	Natural Lan-guage Processing	• A good guide for understanding security risks in software development. • Simple and easy-to-comprehend explanations of concepts.	• Results are not validated with support or numerical data. • Does not cover a complete range of security risks.

(Continued)

Table 9 (continued)

Ref.	Approach/Area covered	Strengths	Weaknesses
[68]	Protection Poker technique	• Introduces a novel idea in the form of Protection Poker. • Includes empirical validation techniques. • Simple and easy-to-understand proposed model.	• Small sample size for empirical validation. • Lacks comparative analysis with other methods.
[82]	Artificial neural networks	• Good combination of soft computing techniques with ANNs for estimating effort in assessing security risks. • Detailed analysis of ANNs with soft computing techniques is provided.	• The proposed model is complex and resource-intensive. • Validation data does not encompass a wide range of software projects.
[74,84,85]	Analysis of machine learning techniques as effort estimation techniques	• Comparative analysis of various ML techniques is conducted. • Identifies hazards that can be mitigated through different techniques.	• Small sample size for empirical validation. • Lacks comparative analysis with other methods.

4.19 Comparative Analysis of Models

Comparing famous security integrating frameworks like OWASP's CLASP, Microsoft SDL, NIST, etc. shows that almost all are effective in a way that helps in mitigating security hazards but in their way. Studies have revealed that among all CLASP is found to be more adjustable according to the changing needs of the organization regarding security [7,32]. Protection poker a risk estimation technique is also useful for indicating and analyzing security threats earliest in the development process [24]. Fig. 7 shows the ratio of studies addressing various aspects of software security.

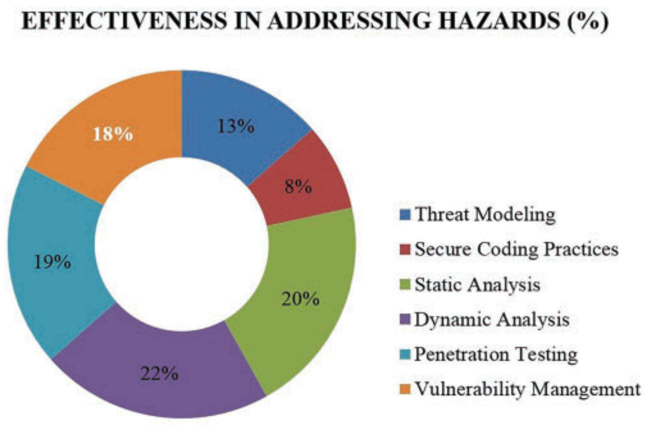

Figure 7: Effective in addressing various hazards

5 Conclusion and Future Works

This review emphasizes the critical role of security throughout the software development life cycle (SDLC). A proactive approach to risk management involving a thorough evaluation of potential threats is required to ensure security. Signification challenges exist in integrating security into SDLC such as fostering cultural change within organizations, meeting needs for diversified skills, and achieving comprehensive security throughout the development process. Organizations must assign security a top priority, and make sure that the employees are skilled enough to practice security measures in day-to-day operations. The review also underlines the prospect of automation in enhancing the accuracy of security practices. Security tools can help in the identification and mitigation of security threats more efficiently. One of the key challenges is the accuracy of the risk estimation method required for ascertaining the priority of potential security hazards. Accuracy can also be improved by exploring new risk estimation techniques involving the use of deep learning and artificial intelligence methods. The authentication of these AI-integrated methods is also critical to ensure the reliable mitigation of potential threats.

Threat analysis and security testing can be explored further to develop automated tools that can pave the way for future research. Researchers should also adopt well-recognized checklists and standards to enhance the robustness and comparability of future studies. Avenues like AI-drive models such as convolutional neural networks and boosting models provide a promising prospect for reducing security vulnerabilities and improving overall organizational security postures. Thus, by progressing in automation and adopting emerging AI technologies, the security frameworks of the organizations can be reinforced and can contribute to a more resilient software environment.

Acknowledgement: None.

Funding Statement: None.

Author Contributions: The authors confirm contributions to the paper as follows: study conception and design: Hassan Saeed, Imran Shafi; data collection: Imran Shafi, Jamil Ahmad; analysis and interpretation of results: Adnan Ahmed Khan, Tahir Khurshaid, Imran Ashraf; draft manuscript preparation: Hassan Saeed, Tahir Khurshaid; manuscript final layout and preparation for submission:

Adnan Ahmed Khan, Jamil Ahmad, Imran Ashraf. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] E. Khanna, R. Popli, and N. Chauhan, "Identification and classification of risk factors in distributed agile software development," *J. Web Eng.*, vol. 21, no. 6, pp. 1831–1851, 2022. doi: 10.13052/jwe1540-9589.2164.

[2] P. Singal, A. C. Kumari, and P. Sharma, "Estimation of software development effort: A differential evolution approach," *Procedia Comput. Sci.*, vol. 167, pp. 2643–2652, 2020. doi: 10.1016/j.procs.2020.03.343.

[3] H. Assal and S. Chiasson, "Security in the software development lifecycle," presented at the Fourteenth Symp. Usable Privacy Secur. (SOUPS 2018), Baltimore, MD, USA, Aug. 12–14, 2018.

[4] R. A. Khan, S. U. Khan, M. Alzahrani, and M. Ilyas, "Security assurance model of software development for global software development vendors," *IEEE Access*, vol. 10, no. 2, pp. 58458–58487, 2022. doi: 10.1109/ACCESS.2022.3178301.

[5] H. Myrbakken and R. Colomo-Palacios, "DevSecOps: A multivocal literature review," in *Software Process Improvement and Capability Determination*, A. Mas, A. Mesquida, R. O'Connor, T. Rout, and A. Dorling, Eds., 1st ed. Palma de Mallorca, Spain: Springer, 2017, vol. 770, pp. 17–29. doi: 10.1007/978-3-319-67383-7_2.

[6] S. Al-Saqqa, S. Sawalha, and H. AbdelNabi, "Agile software development: Methodologies and trends," *Int. J. Interact. Mob. Technol.*, vol. 14, no. 11, pp. 246–270, 2020. doi: 10.3991/ijim.v14i11.13269.

[7] I. El Rhaffari and O. Roudiès, "Benchmarking SDL and CLASP lifecycle," presented at the 9th Int. Conf. Intell. Syst.: Theor. Appl. (SITA-14), Rabat, Morocco, May 14–15, 2014.

[8] V. Sharma, "Enhancing software security through automation in the software development lifecycle," *J. Artif. Intell. Cloud Comput.*, vol. 1, pp. 1–4, 2022. doi: 10.47363/JAICC/2022(1)169.

[9] L. M. Clagett II, "Security requirements for the prevention of modern software vulnerabilities and a process for incorporation into classic software development lifecycles," Ph.D. dissertation, Virginia Tech, Blacksburg, VA, USA, 2009.

[10] M. Otieno, D. Odera, and J. E. Ounza, "Theory and practice in secure software development lifecycle: A comprehensive survey," *World J. Adv. Res. Rev.*, vol. 18, no. 3, pp. 53–78, 2023. doi: 10.30574/wjarr.2023.18.3.0944.

[11] H. Myrbakken and R. Colomo-Palacios, "DevSecOps: A multivocal literature review," in *Software Process Improv. Capability Determination: 17th Int. Conf., SPICE 2017*, Palma de Mallorca, Spain, Oct. 4–5, 2017, pp. 17–29. doi: 10.1007/978-3-319-67383-7_2.

[12] J. Alonso, R. Piliszek, and M. Cankar, "Embracing IaC through the DevSecOps philosophy: Concepts, challenges, and a reference framework," *IEEE Softw.*, vol. 40, no. 1, pp. 56–62, Feb. 2023. doi: 10.1109/MS.2022.3212194.

[13] G. B. White and N. Sjelin, "The NIST cybersecurity framework," in *Research Anthology on Business Aspects of Cybersecurity*. USA: IGI Global, 2022, pp. 39–55. doi: 10.4018/978-1-6684-3698-1.ch003.

[14] K. Qian, R. M. Parizi, and D. Lo, "OWASP risk analysis driven security requirements specification for secure Android mobile software development," presented at the IEEE Conf. Dependable Secure Comput. (DSC), Kaohsiung, Taiwan, Dec. 2018.

[15] OWASP Foundation, "OWASP Top 10," 2021. Accessed: Dec. 04, 2024. [Online]. Available: https://owasp.org/www-project-top-ten/

[16] M. Srivastava, A. Raghuvanshi, and D. Khandelwal, “Security and scalability of e-commerce website by OWASP threats,” presented at the 6th Int. Conf. Inf. Syst. Comput. Netw. (ISCON), Mathura, India, Nov. 2023.

[17] D. Odera, M. Otieno, and J. E. Ounza, “Security risks in the software development lifecycle: A review,” *World J. Adv. Eng. Technol. Sci.*, vol. 8, no. 2, pp. 230–253, 2023. doi: 10.30574/wjaets.2023.8.2.0101.

[18] W. Umeugo, “Secure software development lifecycle: A case for adoption in software SMEs,” *Int. J. Adv. Res. Comput. Sci.*, vol. 14, no. 1, pp. 1–7, 2023. doi: 10.26483/ijarcs.v14i2.6955.

[19] D. Ferdiansyah, R. Isnanto, and J. E. Suseno, “Organizational indicators on startup software for implementing secure software development lifecycle (SSDL): A systematic literature review,” AIP Conf. Proc., vol. 2683, 2023, Art. no. 040001. doi: 10.1063/5.0125388.

[20] X. Meng, K. Qian, D. Lo, P. Bhattacharya, and F. Wu, “Secure mobile software development with vulnerability detectors in static code analysis,” presented at the Int. Symp. Netw., Comput. Commun. (ISNCC), Rome, Italy, Jun. 2018.

[21] G. Jahanavi, T. Mubeen, R. Aishwarya, and R. Yogitha, “Cloud computing using OWASP: Open web application security project,” presented at the 7th Int. Conf. Intell. Comput. Control Syst. (ICICCS), Madurai, India, May 17–19, 2023.

[22] J. Gregoire, K. Buyens, B. De Win, R. Scandariato, and W. Joosen, “On the secure software development process: CLASP and SDL compared,” presented at the Third Int. Workshop Softw. Eng. Secure Syst. (SESS’07: ICSE Workshops 2007), Minneapolis, MN, USA, May 20–26, 2007.

[23] M. Howard and S. Lipner, *The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software*. Redmond, WA, USA: Microsoft Press, 2006.

[24] A. Mohammad, J. Alqatawna, and M. Abushariah, “Secure software engineering: Evaluation of emerging trends,” presented at the 8th Int. Conf. Inf. Technol. (ICIT), Amman, Jordan, May 17–18, 2017.

[25] M. F. Fauzi, V. R. Mohan, Y. Qi, C. Chandrasegar, and S. Muzafar, “Secure software development: Best practices,” *Int. J. Emerg. Multidisciplinaries Comput. Sci. Artif. Intell.*, vol. 2, no. 1, pp. 1–18, 2023. doi: 10.54938/ijemdcsai.2023.02.1.256.

[26] M. Kainerstorfer, J. Sametinger, and A. Wiesauer, “Software security for small development teams: A case study,” presented at the 13th Int. Conf. Inf. Integr. Web-Based Appl. Serv., Ho Chi Minh City, Vietnam, Dec. 5–7, 2011.

[27] S. Modi and D. R. Nijga, “Changing security with every spiral,” *Int. J. Sci. Res. Eng. Manage.*, vol. 7, no. 4, pp. 45–50, 2023. doi: 10.55041/IJSREM20194.

[28] S. Roy and P. S. Mudundi, “Implementing a secure web-based application using Microsoft SDL,” in *Computer and Cyber Security: Principles, Algorithm, Applications, and Perspectives*, A. K. Sangaiah, Ed. Boca Raton, FL, USA: Auerbach Publications, 2018, pp. 355–388.

[29] S. A. Asadollah, “Cyberattacks: Modeling, analysis, and mitigation,” presented at the 6th Int. Conf. Comput., Software Modeling (ICCSM), Barcelona, Spain, Jul. 15–17, 2022.

[30] G. McGraw, “Building Security IN,” *Datenschutz und Datensicherheit-DuD*, vol. 36, no. 9, pp. 662–665, Sep. 2012. doi: 10.1007/s11623-012-0172-4.

[31] K. Tiirik, “Comparison of SDL and Touchpoints,” M.S. thesis, Dept. of Computer Science, Univ. of Tartu, Tartu, Estonia, 2013.

[32] B. De Win, R. Scandariato, K. Buyens, J. Grégoire, and W. Joosen, “On the secure software development process: CLASP, SDL and Touchpoints compared,” *Inf. Softw. Tech.*, vol. 51, no. 7, pp. 1152–1171, Jul. 2009. doi: 10.1016/j.infsof.2008.01.010.

[33] M. -G. Lee, H. Sohn, M. Baek, and J. -B. Kim, “A study on secure SDLC specialized in common criteria,” presented at the Int. Conf. Adv. Sci. Technol. Lett., Jeju Island, Republic of Korea, Dec. 2015.

[34] A. Sharma and R. Bawa, “Identification and integration of security activities for secure agile development,” *Int. J. Inf. Technol.*, vol. 14, no. 2, pp. 1117–1130, Apr. 2022. doi: 10.1007/s41870-020-00446-4.

[35] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey and S. Linkman, “Systematic literature reviews in software engineering-A systematic literature review,” *Inf. Softw. Tech.*, vol. 51, no. 1, pp. 7–15, Jan. 2009. doi: 10.1016/j.infsof.2008.09.009.

[36] C. Onwubiko, "Security operations centre: Situation awareness, threat intelligence and cybercrime," presented at the Int. Conf. Cyber Situat. Aware., Data Analytics Assess. (Cyber SA), London, UK, Jun. 2017.

[37] C. Wohlin, M. Kalinowski, K. R. Felizardo, and E. Mendes, "Successful combination of database search and snowballing for identification of primary studies in systematic literature studies," *Inf. Softw. Tech.*, vol. 147, no. 7, Mar. 2022, Art. no. 106908. doi: 10.1016/j.infsof.2022.106908.

[38] S. Panda, "Reference management software for assisting researchers: A comparative analysis of usage and usability," in *Library Technology with New Perception*, 1st ed. New Delhi, India: ESS Publications, 2023, pp. 191–206.

[39] L. Yang, H. Zhang, H. Shen, X. Huang, and X. Zhou, "Quality assessment in systematic literature reviews: A software engineering perspective," *Inf. Softw. Tech.*, vol. 130, no. 2, Dec. 2021, Art. no. 106397. doi: 10.1016/j.infsof.2020.106397.

[40] J. Shah and N. Kama, "Extending function point analysis effort estimation method for software development phase," presented at the 7th Int. Conf. Softw. Comput. Appl., Kuantan, Malaysia, Feb. 8–10, 2018.

[41] D. Childress, "Citation tools in academic libraries: Best practices for reference and instruction," *Ref. User Serv. Q.*, vol. 51, no. 2, pp. 143–152, Winter 2011. doi: 10.5860/rusq.51n2.143.

[42] M. Alenezi and S. Almuairfi, "Security risks in the software development lifecycle," *Int. J. Recent Technol. Eng.*, vol. 8, no. 3, pp. 7048–7055, Sep. 2019. doi: 10.35940/ijrte.C5374.098319.

[43] W. Wang, Q. Zeng, and A. P. Mathur, "A security assurance framework combining formal verification and security functional testing," presented at the 12th Int. Conf. Qual. Softw., Xi'an, China, Aug. 27–28, 2012.

[44] I. A. Tøndel, M. G. Jaatun, D. S. Cruzes, and L. Williams, "Collaborative security risk estimation in agile software development," *Inf. Comput. Secur.*, vol. 27, no. 4, pp. 508–535, 2019. doi: 10.1108/ICS-12-2018-0138.

[45] S. -J. Chen, Y. -C. Pan, Y. -W. Ma, and C. -M. Chiang, "The impact of the practical security test during the software development lifecycle," presented at the 24th Int. Conf. Adv. Commun. Technol. (ICACT), Pyeongchang, Republic of Korea, Feb. 13–16, 2022.

[46] M. G. Stochel, "Reliability and accuracy of the estimation process—Wideband Delphi *vs.* wisdom of crowds," presented at the 35th Annual Comput. Softw. Appl. Conf., Munich, Germany, Jul. 18–22, 2011.

[47] Y. Zeng, Y. Cheng, G. Xie, and R. Wang, "Design of mobile application lifecycle security management platform," presented at the Int. Conf. Comput. Netw., Electron. Autom. (ICCNEA), Xi'an, China, Dec. 10–12, 2021.

[48] A. I. Al-Darwish and P. Choe, "A framework of information security integrated with human factors," in *HCI Cybersecurity, Priv. Trust: First Int. Conf., HCI-CPT 2019, Part 21st HCI Int. Conf.*, Orlando, FL, USA, Jul. 26–31, 2020, vol. 11594, pp. 39–55.

[49] O. Kovalenko, O. Smirnov, A. Kovalenko, and S. Kavun, "Quantitative risk assessment method development in the context of the SDLC-model," presented at the 8th Int. Conf. Probl. Info Commun., Sci Technol. (PIC S&T), Kharkiv, Ukraine, Oct. 5–8, 2021.

[50] M. U. Nisa, M. Saqlain, A. A. Naeem, M. Awais, and Ž. Stević, "Analysis of software effort estimation by machine learning techniques," *Ing Syst. Inf.*, vol. 28, no. 6, pp. 1445–1457, 2023. doi: 10.18280/isi.280602.

[51] A. S. Farhan and G. M. Mostafa, "A methodology for enhancing software security during development processes," presented at the 21st Saudi Comput. Soc. Natl. Comput. Conf. (NCC), Riyadh, Saudi Arabia, Apr. 25–26, 2018.

[52] S. L. Kanniah and M. N. Mahrin, "Secure software development practice adoption model: A Delphi study," *J. Telecommun., Electronic Comput. Eng.*, vol. 10, no. 2–8, pp. 71–75, 2018.

[53] M. Gutfleisch, M. Schöps, S. A. Horstmann, D. Wichmann, and M. A. Sasse, "Security champions without support: Results from a case study with OWASP SAMM in a large-scale e-commerce enterprise," presented at the 2023 Eur. Symp. Usable Secur., Barcelona, Spain, Jun. 12–14, 2023.

[54] Z. A. Maher, H. Shaikh, M. S. Khan, A. Arbaaeen, and A. Shah, "Factors affecting secure software development practices among developers—An investigation," presented at the 5th Int. Conf. Eng. Technol. Appl. Sci. (ICETAS), Bangkok, Thailand, Nov. 22–23, 2018.

[55] M. Alhamed and T. Storer, "Playing planning poker in crowds: Human computation of software effort estimates," presented at the IEEE/ACM 43rd Int. Conf. Softw. Eng. (ICSE), Madrid, Spain, May 25–28, 2021.

[56] N. Ziems and S. Wu, "Security vulnerability detection using deep learning natural language processing," presented at the IEEE INFOCOM 2021-IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS), Vancouver, BC, Canada, May 10–13, 2021.

[57] P. Kadebu, V. Thada, and P. Chiurunge, "Natural language processing and deep learning towards security requirements classification," presented at the 3rd Int. Conf. Contemp. Comput. Inform. (IC3I), Gurgaon, India, Dec. 27–29, 2018.

[58] C. A. Cois and R. Kazman, "Natural language processing to quantify security effort in the software development lifecycle," presented at the Int. Conf. Softw. Eng. Knowl. Eng. (SEKE), Pittsburgh, PA, USA, Jul. 6–8, 2015.

[59] C. Do Xuan, V. N. Son, and D. Duc, "Automatically detect software security vulnerabilities based on natural language processing techniques and machine learning algorithms," *J. ICT Res. Appl.*, vol. 16, no. 1, pp. 1–10, 2022. doi: 10.5614/itbj.ict.res.appl.2022.16.1.1.

[60] K. Singh, S. S. Grover, and R. K. Kumar, "Cyber security vulnerability detection using natural language processing," presented at the IEEE World AI IoT Congr. (AIIoT), Seattle, WA, USA, May 5–9, 2022.

[61] M. Wang, L. Xu, and L. Guo, "Anomaly detection of software system logs based on natural language processing," presented at the Int. Conf. Image Video Process. Artif. Intell., Honolulu, HI, USA, Jul. 12–15, 2018, vol. 10836.

[62] B. K. Verma and A. K. Yadav, "Software security with natural language processing and vulnerability scoring using machine learning approach," *J. Ambient Intell. Humaniz. Comput.*, vol. 15, no. 4, pp. 2641–2651, 2024. doi: 10.1007/s12652-024-04778-y.

[63] W. Marcellino, R. Harris, K. Galai, K. Cox, L. Slapakova and A. Jaycocks, "Natural language processing: Security-and defense-related lessons learned," RAND Corporation, 2020, Accessed: Dec. 04, 2024. [Online]. Available: https://www.rand.org/pubs/research_reports/RRA519-1.html.

[64] M. A. Khan, S. Khan, and M. Sadiq, "Systematic review of software risk assessment and estimation models," *Int. J. Eng. Adv. Technol.*, vol. 1, no. 4, pp. 243–252, Nov. 2012. doi: 10.1007/s40012-013-0022-4.

[65] I. A. Tøndel, M. G. Jaatun, D. Cruzes, and T. D. Oyetoyan, "Understanding challenges to adoption of the protection poker software security game," presented at the ESORICS 2018 Int. Workshops, Barcelona, Spain, Sep. 6–7, 2018.

[66] A. Olmsted, "Secure software development—Models, tools, architectures and algorithms," presented at the 11th Int. Conf. Emerg. Secur. Inf., Syst. Technol. (SECURWARE), Barcelona, Spain, Oct. 23–27, 2017.

[67] M. G. Jaatun and I. A. Tøndel, "Playing protection poker for practical software security," presented at the Product-Focused Softw. Process Improv.: 17th Int. Conf., PROFES 2016, Trondheim, Norway, Nov. 22–24, 2016.

[68] H. Rygge and A. Jøsang, "Threat poker: Solving security and privacy threats in agile software development," presented at the 23rd Nordic Conf. Secure IT Systems (NordSec 2018), Oslo, Norway, Nov. 28–30, 2018, pp. 468–483.

[69] A. P. Subriadi and N. F. Najwa, "The consistency analysis of failure mode and effect analysis (FMEA) in information technology risk assessment," *Heliyon*, vol. 6, no. 1, Jan. 2020, Art. no. e03153. doi: 10.1016/j.heliyon.2020.e03161.

[70] M. Da Silva, M. Puys, P. -H. Thevenon, S. Mocanu, and N. Nkawa, "Automated ICS template for STRIDE Microsoft threat modeling tool," presented at the 18th Int. Conf. Availab., Reliab. Secur. (ARES), Benevento, Italy, Aug. 28–30, 2023.

[71] L. Mauri and E. Damiani, "Modeling threats to AI-ML systems using STRIDE," *Sensors*, vol. 22, no. 17, Aug. 2022, Art. no. 6662. doi: 10.3390/s22176662.

[72] A. Kumar, "Recommendation of machine learning techniques for software effort estimation using multi-criteria decision making," *J. Univers. Comput. Sci.*, vol. 30, no. 2, pp. 45–58, 2024. doi: 10.3897/jucs.110051.

[73] T. J. Gandomani, K. T. Wei, and A. K. Binhamid, "A case study research on software cost estimation using experts' estimates, wideband Delphi, and planning poker technique," *Int. J. Softw. Eng. Appl.*, vol. 8, no. 11, pp. 173–182, Nov. 2014.

[74] W. S. Ismail, "Threat detection and response using AI and NLP in cybersecurity," presented at the 2020 Int. Conf. Cybersecurity Priv., New York, NY, USA, Jul. 22–25, 2020.

[75] A. H. Salem, S. M. Azzam, O. Emam, and A. A. Abohany, "Advancing cybersecurity: A comprehensive review of AI-driven detection techniques," *J. Big Data*, vol. 11, no. 1, Jan. 2024, Art. no. 105. doi: 10.1186/s40537-024-00957-y.

[76] P. Radanliev *et al.*, "Artificial intelligence and machine learning in dynamic cyber risk analytics at the edge," *SN Appl. Sci.*, vol. 2, no. 11, Apr. 2020, Art. no. 1773. doi: 10.1007/s42452-020-03559-4.

[77] S. Silvestri, S. Islam, S. Papastergiou, C. Tzagkarakis, and M. Ciampi, "A machine learning approach for the NLP-based analysis of cyber threats and vulnerabilities of the healthcare ecosystem," *Sensors*, vol. 23, no. 2, Jan. 2023, Art. no. 651. doi: 10.3390/s23020651.

[78] K. Bennouk, N. Ait Aali, Y. El Bouzekri El Idrissi, B. Sebai, A. Z. Faroukhi and D. Mahouachi, "A comprehensive review and assessment of cybersecuri vulnerability detection methodologies," *J. Cybersecuri Priv.*, vol. 4, no. 4, pp. 853–908, Dec. 2024. doi: 10.3390/jcp4040040.

[79] A. A. Ardo, J. M. Bass, and T. Gaber, "Towards secure agile software development process: A practice-based model," presented at the 48th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA), Porto, Portugal, Sep. 2–5, 2022. doi: 10.1109/SEAA56994.2022.00031.

[80] S. Kassaymeh, M. Alweshah, M. A. Al-Betar, A. I. Hammouri, and M. A. Al-Ma'aitah, "Software effort estimation modeling and fully connected artificial neural network optimization using soft computing techniques," *Cluster Comput.*, vol. 27, no. 1, pp. 737–760, Feb. 2024. doi: 10.1007/s10586-023-03979-y.

[81] S. Bilgaiyan, S. Mishra, and M. Das, "A review of software cost estimation in agile software development using soft computing techniques," presented at the 2nd Int. Conf. Comput. Intell. Netw. (CINE), Bhubaneswar, India, Jan. 23–24, 2016. doi: 10.1109/CINE.2016.27.

[82] W. G. Johnson and C. L. Cheng, "A survey of software effort estimation techniques using machine learning research-oriented," presented at the 10th Int. Conf. Predictive Models Softw. Eng., Denver, CO, USA, Sep. 19–20, 2017.

[83] P. Brar and D. Nandal, "A systematic literature review of machine learning techniques for software effort estimation models," presented at the Fifth Int. Conf. Comput. Intell. Commun. Technol. (CCICT), Ghaziabad, India, Feb. 25–26, 2022, pp. 494–499.

[84] B. Prakash and V. Viswanathan, "A survey on software estimation techniques in traditional and agile development models," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 7, no. 3, pp. 867–876, Mar. 2017. doi: 10.11591/ijeecs.v7.i3.pp867-876.

[85] F. B. Ahmad and L. M. Ibrahim, "Software development effort estimation techniques using long short-term memory," presented at the Int. Conf. Comput. Sci. Softw. Eng. (CSASE), Duhok, Iraq, Apr. 12–13, 2022.

[86] A. B. Farid, E. M. Fathy, A. S. Eldin, and L. A. Abd-Elmegid, "Software defect prediction using hybrid model (CBIL) of convolutional neural network (CNN) and bidirectional long short-term memory (Bi-LSTM)," *PeerJ Comput. Sci.*, vol. 7, no. 5, Dec. 2021, Art. no. e739. doi: 10.7717/peerj-cs.739.

[87] R. Marco, S. S. S. Ahmad, and S. Ahmad, "An improving long short-term memory-grid search-based deep learning neural network for software effort estimation," *Int. J. Intell. Eng. Syst.*, vol. 16, no. 4, pp. 123–130, Aug. 2023. doi: 10.22266/ijies2023.0831.14.

[88] N. Rankovic and D. Rankovic, "Power of LSTM and SHAP in the use case point approach for software effort and cost estimation," presented at the IEEE 22nd World Symp. Appl. Mach. Intell. Inform. (SAMI), Herľany, Slovakia, Jan. 24–26, 2024. doi: 10.1109/SAMI60510.2024.10432878.

[89] R. Naseem *et al.*, “Empirical assessment of machine learning techniques for software requirements risk prediction,” *Electronics*, vol. 10, no. 2, Jan. 2021, Art. no. 168. doi: 10.3390/electronics10020168.

[90] X. Zhu, A. Wang, K. Zhang, and X. Hua, “A deep learning method to mitigate the impact of subjective factors in risk estimation for machinery safety,” *Appl. Sci.*, vol. 14, no. 11, Jun. 2024, Art. no. 4519. doi: 10.3390/app14114519.

[91] M. Awais and A. A. Malik, “A comparative analysis of popular software effort estimation techniques,” presented at the Int. Conf. IT Ind. Technol. (ICIT), Karachi, Pakistan, Mar. 12–14, 2023.

[92] S. Sharma and S. Vijayvargiya, “An optimized neuro-fuzzy network for software project effort estimation,” *IETE J. Res.*, vol. 69, no. 10, pp. 6855–6866, Oct. 2023. doi: 10.1080/03772063.2022.2027282.

[93] F. B. Alhamdany and L. M. Ibrahim, “Software development effort estimation techniques: A survey,” *J. Educ. Sci.*, vol. 31, no. 1, pp. 80–92, Feb. 2022. doi: 10.33899/edusj.2022.132274.1201.

[94] A. Kialbekov, “Empirical study on commonly used combinations of estimation techniques in software development planning,” presented at the Euro. Symp. Softw. Eng. (ESSE 2020), Cambridge, UK, Nov. 5–6, 2020.

[95] V. Chahar and P. K. Bhatia, “Unveiling the art of software testing effort estimation: An in-depth study of current techniques and their analysis,” presented at the Int. Conf. Adv. Comput. Commun. Technol. (ICACCTech), Dehradun, India, Jan. 15–17, 2023.

[96] J. Lande, N. Tejaswini, H. M. Al-Jawahry, P. Lekshmy, and M. Sridevi, “Automatic software vulnerability analysis through a multi-head attention and long short-term memory approach,” presented at the Int. Conf. Integr. Circuits Commun. Syst. (ICICACS), Hyderabad, India, Mar. 6–8, 2024.

[97] M. A. Mateen and A. A. Malik, “A comparative study of the accuracy and efficiency of wideband Delphi and planning poker software effort estimation techniques,” presented at the Int. Conf. IT Ind. Technol. (ICIT), Karachi, Pakistan, Mar. 12–14, 2023.

[98] F. Calefato and F. Lanubile, “A planning poker tool for supporting collaborative estimation in distributed agile development,” presented at the 6th Int. Conf. Softw. Eng. Adv. (ICSEA), Barcelona, Spain, Oct. 23–28, 2011.

[99] F. Osses, G. Márquez, C. Orellana, and H. Astudillo, “Towards the selection of security tactics based on non-functional requirements: Security tactic planning poker,” presented at the 36th Int. Conf. Chil. Comput. Sci. Soc. (SCCC), Arica, Chile, Oct. 16–20, 2017, pp. 1–8. doi: 10.1109/SCCC.2017.8405144.

[100] V. Resmi and K. Anitha, “Software effort estimation using machine learning techniques,” *SAMRIDDHI: J Phys. Sci. Eng. Technol.*, vol. 15, no. 1, pp. 86–90, 2023. doi: 10.18090/10.18090/samriddhi.v15i01.12.