Security risks of global software development life cycle: Industry practitioner's perspective

Khan Rafiq Ahmad, Khan Siffat Ullah, Akbar Muhammad Azeem, Alzahrani Musaad

DOI: 10.1002/smr.2521

Security Risks of Global Software Development Life Cycle: Industry Practitioner's Perspective

Rafiq Ahmad Khan[1], Siffat Ullah Khan[1], Muhammad Azeem Akbar[2], Musaad Alzahrani[3]

[1]Software Engineering Research Group, Department of Computer Science & IT, University of Malakand, Pakistan
[2]Software engineering, LUT University, Finland.
[3]Department of Computer Science, Albaha University, Albaha 65799, Saudi Arabia

Corresponding author: (rafiqahmadk@gmail.com, azeem.akbar@lut.fi)

Abstract
Software security has become increasingly important since hacking and other attacks on computer systems have grown in popularity in the last few years. As a result, several researchers have examined security solutions as early as the requirement engineering phase. With the growth of the software business and the internet, there is a need to understand the security risks against each phase of the software development life cycle (SDLC). This study aims to empirically investigate and prioritize the risks that could negatively impact the software security aspects of SDLC in the context of global software development (GSD). To achieve the study objectives, we conducted an industrial empirical study to determine the impact of software security threats against each phase of SDLC. Furthermore, the fuzzy analytical hierarchy process (FAHP) was used to prioritize the list of software security risks against the SDLC. The results and analysis of this study provide a ranked-based decision-making framework, which assists the practitioners in considering the most critical security risks on priority. The results show, "improper plan for secure requirement identification, inception, authentication, authorization, and privacy", "lack of threat models updating", "lack of output validation", "lack of certification in the final release and archive", and "spoofing" are declared as top ranked security risks of SDLC in GSD. In addition, the application of FAHP is novel in this domain as it is helpful to address multicriteria decision-making problems.

Keywords: Software Security, Security Risks, Software Development Life Cycle (SDLC), Secure Software Engineering, Fuzzy Analytical Hierarchy Process (FAHP)

1. Introduction

One of the most critical issues facing companies is implementing and managing security in the software development lifecycle (SDLC) [1, 2]. A set of software security standards, guidelines, practices, and certifications can be used to assist in the creation of secure software applications [3]. However, despite the widespread understanding of the importance of presenting scenarios covering the entire SDLC of secure software development, only a few have been documented [4]. Understanding Secure Software Engineering (SSE) methodologies are becoming increasingly important in tackling the problem's technological and psychological components [2]. Secure software engineering (SSE) is the process of designing, building, and testing software so that it becomes secure; this includes secure SDLC processes and secure software development (SSD) methods [5-7]. To produce secure software, one must adhere to the following four steps: software requirements security, design security, implementation security, and testing security [3, 8]. This process strives to strengthen security requirements, implement threat modeling approaches during software design, and adhere to best security practices when coding, reviewing code, and performing testing [6]. This process must be constantly updated to ensure that software products are secure; thus, research indicating the trends in methods, notations, tools, and techniques is needed [1].

1.1 Rational for the Review in the Context of Existing Knowledge

Contrarily, misusing software can lead to significant financial losses, sabotage in the communications industry, data theft in databases, and even human life-threatening software abuse in missile control systems [9, 10]. Khan [11] stated that security concerns heavily influence software quality as software development grows more complicated, distributed, and concurrent. Insecure software affects an organization's reputation with customers, partners, and investors; it raises expenses, as enterprises are obliged to patch unreliable programs; and it delays other development efforts as scarce resources are assigned to address present software problems [11]. The lack of prioritizing security is one of the major causes of widespread vulnerabilities [8]. Even the most conscientious organizations use the "fix and penetrate" strategy in which security is accessed after a project is completed [8]. The disadvantage is that the users do not apply the patches themselves. Aside from that, attackers may devise strategies to exploit new security flaws [12]. Much money was invested in traditional security methods, mainly focused on network systems. They primarily include IDS (Intrusion Detection System), firewalls, encryption, antivirus, and antispyware protection [5, 13].

Although identifying software security threats and implementing secure SDLC techniques is critical, little work has been done on creating secure software development tools, models, and standards [4, 14]. Our previous published study explored the software security risks and their practices in the SDLC phases [2].

1.2 Research Goals and Questions

The following goals are being pursued in the current study: (1) We conduct a questionnaire survey to gather information from global software development experts (researchers and practitioners alike) on the security risks in the SDLC phases. (2) Using the fuzzy analytical hierarchy process (FAHP), rank the investigated security risks regarding their importance to secure SDLC phases in the context of GSD.

However, no research has prioritized the security risks of SDLC phases in the context of global software development. To address software security issues in the context of GSD, we applied the FAHP technique. We believe that a thorough awareness of software security threats throughout the SDLC phases will aid GSD organizations in implementing required secure software development modifications effectively and efficiently. The ranking of security risks helps GSD organizations prioritize the most critical security concerns necessary for successfully implementing software security activities in the context of GSD. This study's goals are met through the following research questions:

RQ1: What are the most cited software security risks in the SDLC phases in the domain of GSD?
RQ2: What is the best way to prioritize the significant software security risks that have been identified?
RQ3: What would be the decision-making framework for the critical software security risks?

The rest of the paper reads as follows: Section 2 provides an overview of the research. Section 3 goes into detail about the study's methodology. The results of the study can be found in Section 4. Results evaluation and analysis are provided in Section 5. Section 6 investigates the study's limitations. Section 7 provides the Implications of the Study. Section 8 provides a conclusion and directions for future research.

2. Background and Motivation

Software security is the concept of creating software that continues to function even when it is attacked maliciously [10, 15]. The best strategy to eliminate software bugs/vulnerabilities is to incorporate security and non-functional specifications into all phases of the SDLC [2, 16]. There has been considerable research on "high integrity" throughout the years, and scholars and practitioners have worked diligently to

produce secure software systems. This section addresses the various approaches to integrating security into the SDLC phases, as well as the security techniques that are frequently utilized in these approaches:

- CMMI, Microsoft Software Development Life Cycle (MS-SDL), Misuse case modeling, Abuse case modeling, Knowledge Acquisition for Automated Specification, System Security Engineering-Capability Maturity Model (SSE-CMM), OWASP, and Secure Tropos Methodology[16].
- McGraw [17, 18] recommends seven touchpoint operations (Abuse cases, Security requirements, Architectural risk analysis, code review and repair, Penetration testing, and security operations) for creating secure software, all of which are connected to software development artifacts.
- Sodiya [19] developed the Secure Software Development Model (SSDM), which provides training to stakeholders in software development with adequate security education.
- Al-Matouq et al. [12] designed a framework Secure Software Design Maturity Model (SSDMM), and the results show that SSDMM helps measure the maturity level of software development organizations.
- Flechas et al. [20] developed AEGIS (Appropriate and Effective Guidance for Information Security), first evaluating device assets and their relationships, then moving on to risk analysis, which defines weaknesses, threats, and risks.
- Gupta et al. [21] developed Team Software Process for Secure Software Development (TSP) specifically for software teams to help them create a high-performance team and prepare their work to produce the best results.
- The Software Engineering Institute (SEI) at Carnegie Mellon University developed the Capability Maturity Model Integration (CMMI) [22] process model, which assists companies measure and improving their development processes while also delivering high-quality products.
- Al-Qutaish and Abran [23] proposed the Software Product Quality Maturity Model (SPQMM), which measures the quality of a software product.
- B. Golden, [24] introduced the Open Source Maturity Model (OSMM) to assess open-source products.
- April et al. [25] proposed the Software Maintenance Maturity Model (SMmm), based on the CMMI, to assess and improve the quality of software maintenance activities.
- Turetken et al. [26] developed a maturity model to assess the Scaled Agile Framework (SAFe), which integrates agile software development practices in traditional large-scale projects.
- Da Silva and de Barros [27] presented an information security maturity model for software developers based on ISO 27001; it was evaluated by subject experts and utilized to measure the maturity level of several organizations.
- S. R. Ahmed [28] identified security activities that should be performed to build secure software and has shown how the security activities are related to usual activities in different phases of software development.
- Essafi et al. [29] developed the Secure Software Development Process Model (S2D-ProM), a strategy-oriented process model that offers guidance and support to developers and software engineers at all levels, from beginners to experts to build secure software.
- Niazi et al. [8] conducted a systematic literature review (SLR) to pinpoint the required practices for developing secure software and identifying best requirement practices. A framework for secure requirement engineering named Requirements Engineering Security Maturity Model (RESMM) was developed.
- Manico [30] designed the Comprehensive, Lightweight Application Security Process (CLASP), which consists of 24 high-level security activities that can be entirely or partially integrated into software during the SDLC.
- The Open Web Application Security Project (OWASP) created the Software Assurance Maturity Paradigm (SAMM) [30], which is a non-commercial model and is an open platform that aids software companies in developing and implementing software security policies.

- The Building Security In Maturity Model (BSIMM) [31] quantifies numerous businesses' security activities and provides a common foundation for them to compare their security endeavours to those of others. There are 119 activities in the BSIMM 10 software security framework. These activities are divided into twelve practices. Each practice's exercises are divided into three maturity levels.
- Security Quality Requirements Engineering (SQUARE) methodology allows for elicitation, classification, and prioritization of security specifications for information technology systems and applications [32].

We conclude from the above discussion that none of these models or structures is explicitly committed to recognizing security risks/threats in the SDLC context of GSD. Furthermore, no study identified and prioritized software security issues during the SDLC phases. An empirical study that analyses security threats and ranks them according to their importance for secure SDLC in GSD is required to fill this research gap.

The study results give a taxonomy that will assist the GSD organizations in developing secure software by developing new and effective strategies to handle the security aspects of SDLC. Apart from that, the knowledge gained from identifying security risks will aid in developing a generic model that would be useful to GSD organizations in effectively implementing security checks against each phase of SDLC. To address the objective of this study, we collected the industry practitioners' insights using a questionnaire survey approach.

Additionally, we apply an expert opinion approach to quantify the discovered security risks against each phase of SDLC. The discovered security risks and their categories were prioritized by calculating the relative weight of each important security risk and its categories. As a result, determining the significance and priority of numerous security risks and categories may be referred to as the multi-criterion decision-making (MCDM) dilemma [33-36]. Numerous decision-making techniques exist [37-42]. Thus, we applied the fuzzy AHP technique as it is an effective way to handle the MCDM. Considering the successfulness of fuzzy AHP in other engineering domains, we also consider it to prioritize the identified security risks against each phase of SDLC.

3. Research Methodology

To achieve the objective of this study, the research work is designed in three steps. A brief description of all the steps is given in subsequent sections:

3.1 Step-1: Identifying software security risks of SDLC in the context of global software development

To explore the software security risks of SDLC in the context of global software development, we used a systematic literature review (SLR) method, and the preliminary findings were published in prior work [2]. An SLR is a secondary study in which primary studies are examined impartially and iteratively to define, interpret, and discuss evidence relevant to the research questions [43-46]. The step-by-step directions of Kitchenham and Charters [44] were followed to conduct the SLR. According to Kitchenham[47], SLR findings are more valid and exhaustive since they are conducted according to predefined protocols. To investigate software security risks during the SDLC phases, Khan et al. [2] thoroughly follow all the SLR processes, namely planning the review, conducting the review, and reporting the review.

One hundred twenty-one papers were selected via the tollgate technique [48] based on the inclusion, exclusion, and quality rating criteria. Khan et al. [2] identified 145 security risks and 424 best practices that help software development organizations to manage security throughout the SDLC phases. Afzal et al. [48] suggested the tollgate method was used to refine the research articles found during the primary study collection.

3.2 Step-2: Empirical Investigation

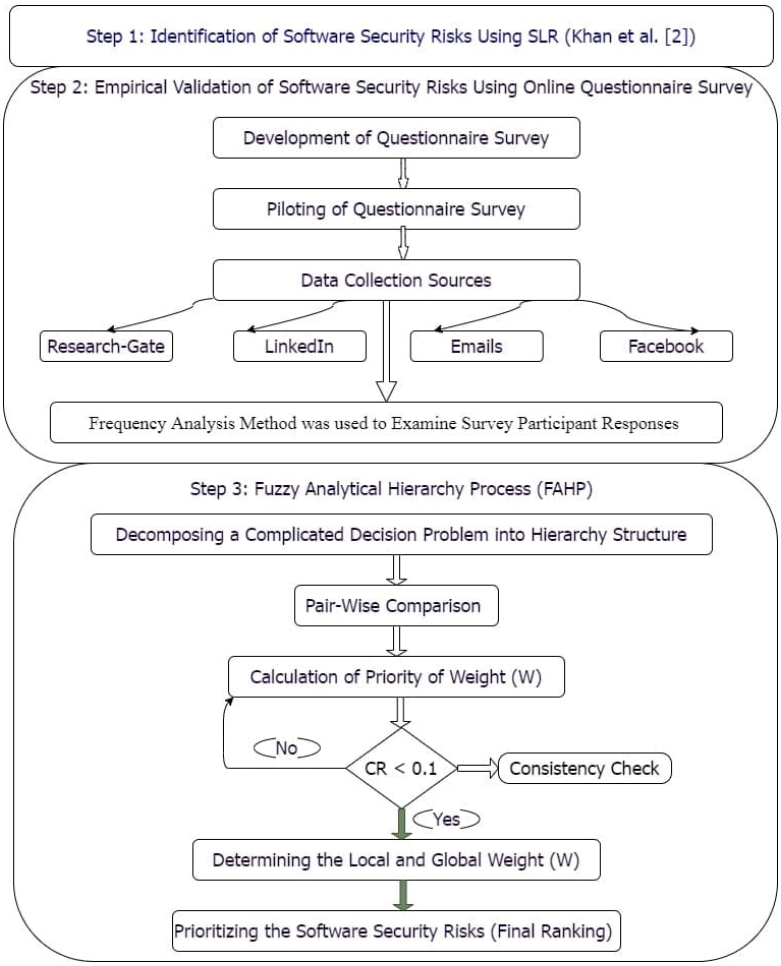

Figure 1: Flowchart of Research Methodology

A key goal of empirical research in software engineering is to assess practical significance, which answers whether the observed effects of some compared treatments show a relevant difference in practice in realistic scenarios. Even though plenty of standard techniques exist to assess statistical significance, connecting it to practical significance is not straightforward or routinely done; only a few empirical studies in software engineering assess practical importance conscientiously and systematically [49].

Many empirical research articles have been published recently [49-53] to address the software security issues. To address this objective, an online questionnaire survey was constructed using Google Docs to validate the SLR findings and discover other security risks and associated practices.

It is difficult to obtain data directly from large numbers of industry experts working across the globe. As a result, we used a non-methodical technique for data collecting, namely an online survey using snow balling technique. Other researchers in the software engineering domain also employed the same

date collection process [54-60]. The following steps were involved in conducting the questionnaire survey:

3.2.1 Development of Questionnaire Survey

The questionnaire primarily consists of closed-ended questions designed to get the practitioner's insight concerning the security risks of SDLC in the GSD context. The questionnaire also contains an open-ended section to allow the survey participant to add any additional security risk of SDLC in GSD. We employed a five-point Likert (strongly-agree to strongly-disagree)scale to obtain survey participants' observations regarding the software security risks and practices listed in the closed-ended section, i.e., strongly agree, agree, neutral, disagree, and strongly disagree.

3.2.2 Pilot of Questionnaire Survey

To conduct the pilot assessment of the questionnaire survey, we chose experts working in the GSD environment (i.e., Software Engineering Research Group (SERG UOM) Pakistan, King Fahd University of Petroleum and Minerals, Saudi Arabia, and Qatar University, Doha, Qatar). This pilot assessment aims to address significant issues (in terms of statistical variables) and improve the survey questions' understandability. Experts suggest improving the questionnaire's design by adding questions to obtain more information about survey participants. The questionnaire survey was revised after considering the experts' ideas and recommendations.

At the beginning of the survey, a statement on the researchers' ethical responsibility was also added to assure the participant's confidentiality. This remark reassured the participants that only the study team would access their information. It was stated that the research team would not share the data with anyone to reveal the identity of any participant or organization.

3.2.2 Data Collection Sources

As previously indicated, our target population was large and spread organizations across the globe. We decided to use unusual methods to collect responses from SSD professionals working in GSD. We used the snowball sampling technique to gather data from the experts [61]. Snowballing is a low-cost, straightforward strategy to reach a specific audience [57, 61, 62].

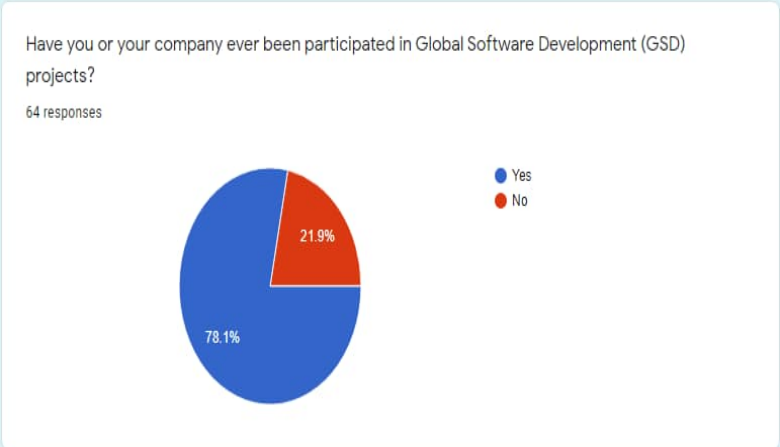

FIGURE 2. Respondents Responses

We used social media networks such as Facebook, LinkedIn, Research Gate, and email to contact the experts. The empirical study's data was collected online from June 01, 2021, to July 04, 2021, and the entire data gathering process took one month and four days. During the survey's implementation, 64 responses were collected, as shown in Figure 2. All of the responses were manually reviewed. We excluded 14 responses because the respondents of these 14 persons do not have experience in GSD and software security. For analysis, the final 50 survey results were taken into account. We ensure survey participants that the obtained data will only be used for research purposes and that their identities will never be disclosed to a third party. Appendix A lists all of the countries and their responses.

3.2.4 Data Analysis

The frequency analysis method was used to examine survey participant replies in this study. This approach analyzes nominal and ordinal data over many variables or groups of variables [63]. Because the survey responses are nominal, we employed the chi-square ("liner-by-linear connection") technique to discover significant differences across the variables. Research with similar data types has used the same analysis approach [62, 64].

3.4 Step-3: Fuzzy Analytical Hierarchy Process (FAHP) Survey

The respondents to the first survey were contacted and asked to participate in a second survey that used the FAHP technique to rank software security risks and their categories. We obtained 27 complete responses from respondents in the second survey. Appendix A contains an example of the questionnaire used to collect data for the second survey. Compared to the previous survey, the FAHP survey had a smaller sample size (27 replies), threatening the generalizability of our results. The FAHP method, on the other hand, has been viewed as a more subjective approach that allows for smaller sample size [65-68].

3.4.1 Fuzzy Set Theory and AHP

This section discusses the basics of fuzzy set theory and how to use it in the traditional AHP method. Several MCDM approaches include AHP, Fuzzy AHP, Fuzzy TOPSIS, etc. AHP is the most popular because it is very effective [33, 69, 70]. Various areas, including political, economic, and management sciences, have extensively used AHP to solve complicated problems. When measuring multiple criteria's relative importance, classical AHP cannot handle the ambiguity and obscurity of the decision-maker. Because of this, fuzzy AHP was developed, which outperformed AHP in terms of accuracy and efficiency [71-73]. With these insights in mind, we chose fuzzy AHP over other approaches.

3.4.2 Fuzzy Set Theory

Zadeh [74] developed a fuzzy set theory to extend the traditional set theory. It was designed to manage ambiguous responses in decision-making tasks based on numerous criteria to deal with uncertainties and ambiguity in real-world situations. The fuzzy set theory is a valuable tool [75, 76]. A characteristic function $\mu_V(x)$ is inserted into the fuzzy set, which maps a given value's membership between 0 and 1. The following sections describe the fundamental principles and definitions of fuzzy sets:

Definition: Triangular fuzzy number (TFN) Vis represented by a triplet (v^l, v^m, v^u). The characteristic function $\mu_V(x)$ of a TFN is given in equation (1) and Figure 3.

$$\mu_V(x) = \begin{cases} \dfrac{x - v^l}{v^m - v^l}, & v^l \le x \le v^m \\ \dfrac{v^u - x}{v^u - v^m}, & v^m \le x \le v^u \\ 0, & Otherwise \end{cases} \qquad (1)$$

Where v^l represents the lowest, v^m represents the most favourable, and v^u denotes the highest possible values.

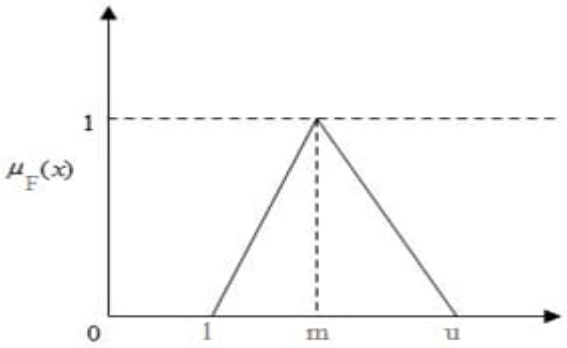

Figure 3. Triangular fuzzy number

Table 1 lists the most typically used algebraic procedures between two TFNs (V_1, V_2).

Table 1. Triangular fuzzy numbers

Operation Law	Expression
Addition ($V_1 \oplus V_2$)	$(v^l_1, v^m_1, v^u_1)_\oplus(v^l_2, v^m_2, v^u_2) = (v^l_1 + v^l_2,\ v^m_1 + v^m_2,\ v^u_1 + v^u_2)$
Subtraction ($V_1 \oplus V_2$)	$(v^l_1, v^m_1, v^u_1)_\oplus(v^l_2, v^m_2, v^u_2) = (v^l_1 - v^l_2,\ v^m_1 - v^m_2,\ v^u_1 - v^u_2)$
Multiplication ($V_1 \oplus V_2$))	$(v^l_1, v^m_1, v^u_1)_\oplus(v^l_2, v^m_2, v^u_2) = (v^l_1 * v^l_2,\ v^m_1 * v^m_2,\ v^u_1 * v^u_2)$
Division ($V_1 \oplus V_2$)	$((v^l_1, v^m_1, v^u_1)_\oplus(v^l_2, v^m_2, v^u_2) = (v^l_1 / v^l_2,\ v^m_1 / v^m_2,\ v^u_1 / v^u_2)$
Inverse ($V_1 \oplus V_2$)	$(v^l_1, v^m_1, v^u_1)^{-1} = (1/v^l_1,\ 1/v^m_1,\ 1/v^u_1)$
For any real number k (kV_1)	$k(v^l_1, v^m_1, v^u_1) = (k\,v^l_1, k\,v^m_1, kv^u_1)$

3.4.3 Fuzzy analytical hierarchy process (FAHP)

Practitioners widely accept that the FAHP is useful for dealing with complex decision-making situations. The most critical aspect of FAHP is its ability to efficiently deal with qualitative and quantitative data that contain many criteria. The significant steps for performing FAHP analysis are as follows:

Step 1: The decision problem is broken down into a hierarchical structure. (See Figure 3)
Step 2: The priority vector is calculated by pair-wise comparison for each level of the hierarchy.
Step 3: Calculate the pair-wise consistency ratio.
Step 4: Prioritize each component and sub-factor based on their weight in the overall score. (See Figure 4).

Even though different FAHP approaches are accessible in the literature [65-68], we have used Chang's method [77] because of its effectiveness and widespread acceptance among scientists. Chang [77] expressed a prioritisation problem as a group of elements referred to as primary categories as X = $\{x_1, x_2,...,x_m\}$. Each x_i also contains elements, called goal set, and represented as V = $\{v_1, v_2,..,v_n\}$. At a time, one main category, x_i, is considered, and each goal g_i undergoes extent analysis. The following equations(2) and(3) can be used to calculate the number of extent analyses (m) that are performed for each category:

$$V^{1}_{gi}, V^{2}_{gi}, \ldots, V^{m}_{gi}, \qquad (2)$$

$$i = 1, 2, \ldots, n \qquad (3)$$

Where, all $V^{j}_{gi,}$ (j = 1, 2, ..., m) are TFNs. Chang's step-by-step extent analysis method is described as follows [77]:

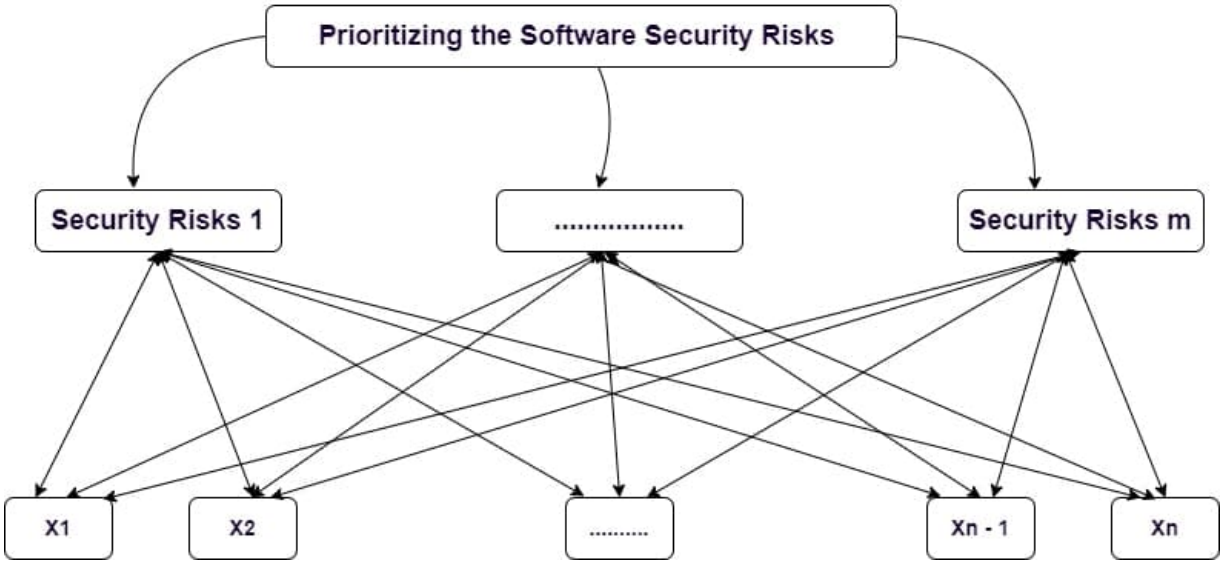

Figure 4. FAHP decision hierarchy

Step 1: The fuzzy analysis of the i^{th} category is shown in Equation (4) as:

$$L_i = \sum_{j=1}^{m} V^{j}{}_{gi} \otimes \left[\sum_{i=1}^{n} \sum_{j=1}^{m} V^{j}{}_{gi} \right]^{-1} \qquad (4)$$

Where $\sum_{j=1}^{m} V^{j}_{gi}$ can be calculated as :

$$\sum^{m} V^{j}{}_{gi} = \left(\sum^{m} v^{l}{}_{gi}, \sum^{m} v^{m}{}_{gi}, \sum^{m} v^{u}{}_{gi}\right) \qquad (5)$$

and $\left[\sum_{i=1}^{n}\sum_{j=1}^{m} V^{j}_{gi}\right]^{-1}$ can be calculated, as mentioned in equation (6) and (7):

$$\sum_{i=1}^{n}\sum_{j=1}^{m} V^{j}{}_{gi} = \left(\sum_{i=1}^{n} v^{l}{}_{i}, \sum_{i=1}^{n} v^{m}{}_{i}, \sum_{i=1}^{n} v^{u}{}_{i}\right) \qquad (6)$$

$$\left[\sum_{i=1}^{n}\sum_{j=1}^{m} V^{j}{}_{gi}\right]^{-1} = \left(\frac{1}{\sum_{i=1}^{n} v^{u}{}_{i}}, \frac{1}{\sum_{i=1}^{n} v^{m}{}_{i}}, \frac{1}{\sum_{i=1}^{n} v^{l}{}_{i}}\right) \qquad (7)$$

Step 2: Given two TFNs V_a and V_b, the degree of possibility that $V_a \geq V_b$ can be defined as:

$$V(V_a \geq V_b) = sup[min(\mu_{Va}(x), (\mu_{Vb}(x))] \qquad (8)$$

Or more specifically as:

$$V(V_a \geq V_b) = hgt(V_a \cap V_b) = \mu_{V_a}(d) = \begin{Bmatrix} 1 & if\ v^{m}{}_{a} \geq v^{m}{}_{b} \\ \dfrac{v^{u}{}_{a} - v^{l}{}_{b}}{(v^{u}{}_{a} - v^{m}{}_{a}) + (v^{m}{}_{b} - v^{l}{}_{b})} & v^{l}{}_{b} \leq v^{u}{}_{a} \\ 0 & Otherwise \end{Bmatrix} \qquad (9)$$

Where d indicates the largest value of intersection between μ_{Ga} and μ_{Vb}(Figure 5)

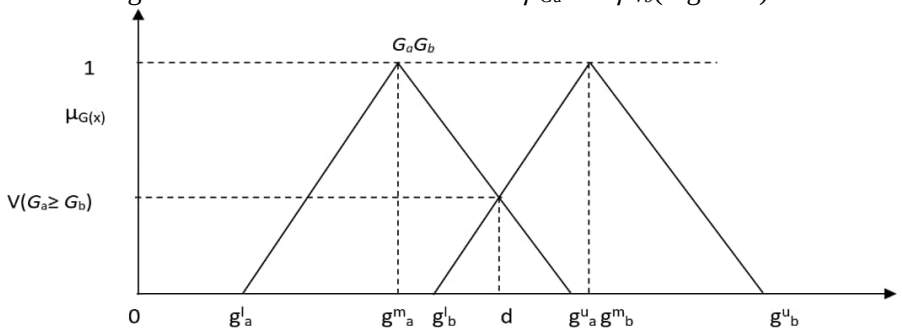

Figure 5. Triangular Fuzzy number

Step 3: **T**he overall degree of possibility of a given convex fuzzy number H is calculated concerning other V_i (i= 1, 2,..., k) as:

$$H(V \geq V_1, V_2, V_3 \ldots V_k) = \min H(V \geq V_i) \qquad (10)$$

Assuming that,

$$d'(V_i) = \min H(V_i \geq V_k) \qquad (11)$$

Where k = 1,2,...,n and $k \neq i$.

The weight vector can be calculated using Eq. 11 as:

$$W' = (d'(V_1), d'(V_2), d'(V_3), \ldots . d'(V_n)) \qquad (12)$$

Where, $V_i (i = 1,2,...,n)$ are n separate fuzzy numbers.

Step 4: The weight vector W obtained from equation (12) is normalized to achieve priority weight as a crisp number:

$$W = (d(V_1), d(V_2), d(V_3), \ldots . d(V_n)) \qquad (13)$$

here, W represents a crisp number.

Step 5: Checking consistency ratio: FAHP requires that the pair-wise matrices be consistent at all times [39]. Thus, a consistency ratio is calculated for each pair of comparison matrices, referred to as a consistency check. A graded mean integration approach converts the given matrix with a fuzzy number into corresponding crisp values. This is referred to as defuzzification. To convert a given TFN P = (l, m, u) into an equivalent crisp number, the following formula must be utilized:

$$P_{crisp} = \frac{(4m + l + u)}{6} \qquad (14)$$

Once P_{crisp} is calculated, the consistency index (CI) and the consistency ratio (CR) is computed as:

$$CI = \frac{I_{max} - n}{n - 1} \qquad (15)$$

$$CR = \frac{CI}{RI} \qquad (16) .$$

Where,
I_{max}: the maximum eigenvalue of the given comparison matrix,
n: number of criteria in the given matrix,
RI: the random index and its value can opt from Table 2.

Table 2. Random consistency index (RI) concerning matrix size

Size of the matrix	1	2	3	4	5	6	7	8	9	10
Random consistency index (RI)	0	0	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49

If the estimated value of CR is less than 0.1, expert pair-wise responses are assumed to be consistent. Otherwise, current responses are discarded, and new responses are gathered.

4. Results and Applications of Fuzzy AHP

To address the research questions raised in Section 1, we have organized the findings of this study into the following sections:

4.1 Most cited software security risks in the SDLC phases in the domain of GSD

We previously published [2] an in-depth analysis of software security risks in the SDLC phases in the domain of Secure Software Engineering (SSE). In this paper, we employed the coding system of Strauss' [78] ground theory (GT) technique to identify, classify, and organize the identified critical security risks (CSRs) in our research. Although we have already collected the data through SLR, we used the four main phases of the GT coding scheme to map the CSRs into four major categories (i.e. "code," "categories", "sub-categories," and theory/theoretical model").

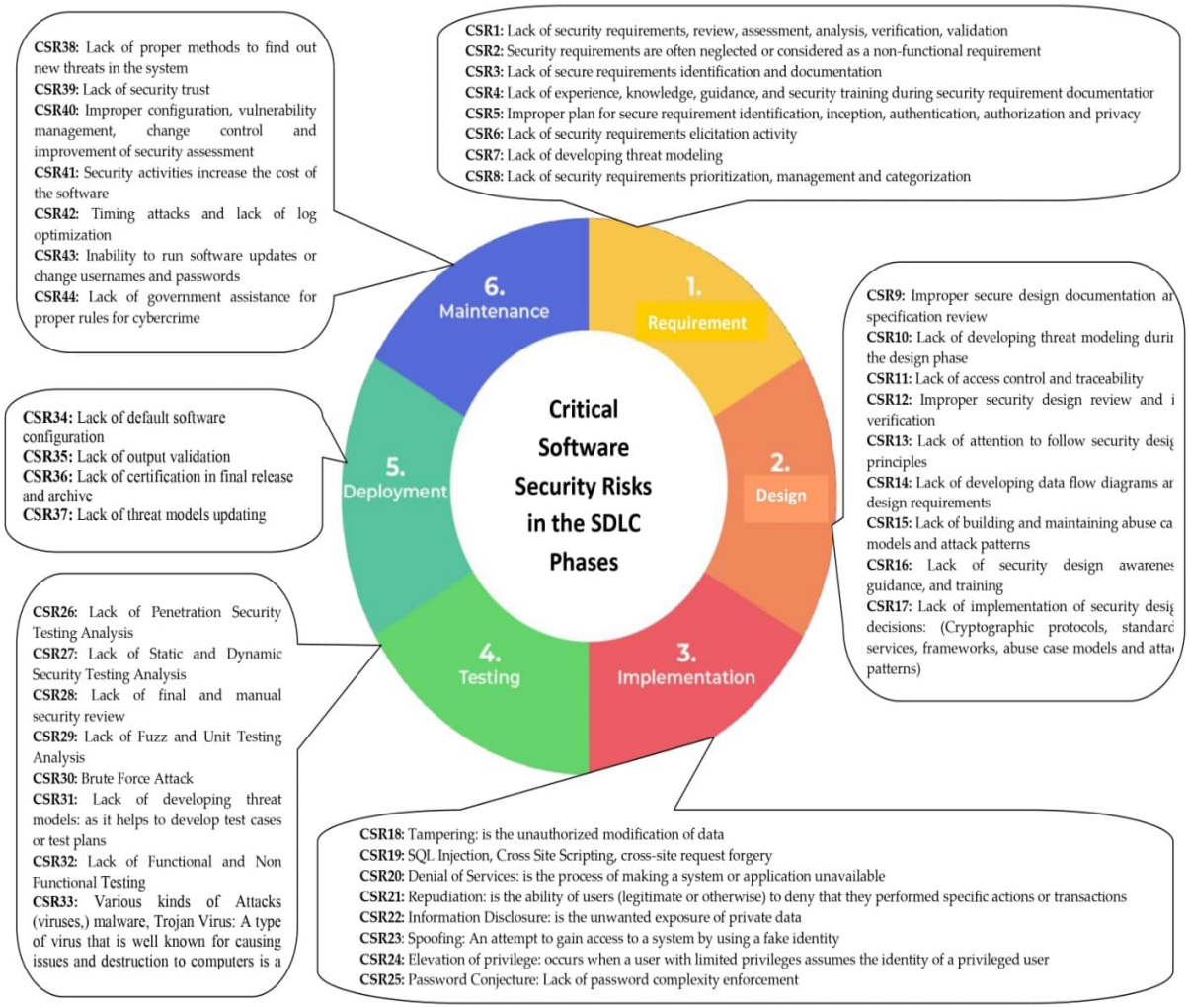

Figure 6: Theoretical model of the critical software security risks of SDLC in GSD

All three authors of this study are on the mapping team. The third author checks to make sure the mapping process is going well. First, we allocated a unique code/label to each CSR we studied. The second phase involved categorizing the examined CSRs into six broad phases/groups, "Requirement Engineering", "Designing", "Coding", "Testing", "Deployment", and "Maintenance". In the third step, the CSRs were mapped into these phases. In the fourth step, we engineered a theoretical model depicted in Figure 6.

The primary goal of this categorization is to construct a hierarchical framework for executing the FAHP. Furthermore, this categorization will help academic researchers and practitioners to identify the most important software security risk in the SDLC in the context of GSD. We identified 45 CSRs, which were mapped in each phase of the SDLC, as stated in Table 3.

Table 3: list of identified critical security risks of SDLC in GSD

SDLC Phases	Code / Label	Critical Security Risks (CSRs)
Requirement Engineering	CSR1	Lack of security requirements, review, assessment, analysis, verification, validation
	CSR2	Security requirements are often neglected or considered a non-functional requirement
	CSR3	Lack of secure requirements identification and documentation
	CSR4	Lack of experience, knowledge, guidance, and security training during security requirement documentation
	CSR5	Improper plan for secure requirement identification, inception, authentication, authorization, and privacy
	CSR6	Lack of security requirements elicitation activity
	CSR7	Lack of developing threat modeling
	CSR8	Lack of security requirements prioritization, management, and categorization
Designing	CSR9	Improper secure design documentation and specification review
	CSR10	Lack of developing threat modeling during the design phase
	CSR11	Lack of access control and traceability
	CSR12	Improper security design review and its verification
	CSR13	Lack of attention to following security design principles
	CSR14	Lack of developing data flow diagrams and design requirements
	CSR15	Lack of building and maintaining abuse case models and attack patterns
	CSR16	Lack of security design awareness, guidance, and training
	CSR17	Lack of implementation of security design decisions: (Cryptographic protocols, standards, services, frameworks, abuse case models, and attack patterns)
Coding	CSR18	Tampering: is the unauthorized modification of data
	CSR19	SQL Injection, Cross Site Scripting, cross-site request forgery
	CSR20	Denial of Services: is the process of making a system or application unavailable
	CSR21	Repudiation: is the ability of users (legitimate or otherwise) to deny that they performed specific actions or transactions
	CSR22	Information Disclosure: is the unwanted exposure of private data
	CSR23	Elevation of privilege: occurs when a user with limited privileges assumes the identity of a privileged user
	CSR24	Spoofing: An attempt to gain access to a system by using a fake identity
	CSR25	Password Conjecture: Lack of password complexity enforcement
Testing	CSR26	Lack of Penetration Security Testing Analysis
	CSR27	Lack of Static and Dynamic Security Testing Analysis
	CSR28	Lack of final and manual security review

	CSR29	Lack of Fuzz and Unit Testing Analysis
	CSR30	Brute Force Attack
	CSR31	Lack of developing threat models: as it helps to develop test cases or test plans
	CSR32	Lack of Functional and Non-Functional Testing
	CSR33	Various kinds of Attacks (viruses,) malware, Trojan Virus: A type of virus that is well known for causing issues and destruction to computers is a Trojan virus
Deployment	CSR34	Lack of default software configuration
	CSR35	Lack of output validation
	CSR36	Lack of certification in final release and archive
	CSR37	Lack of threat models updating
Maintenance	CSR38	Lack of proper methods to find out new threats in the system
	CSR39	Lack of security trust
	CSR40	Improper configuration, vulnerability management, change control, and improvement of security assessment
	CSR41	Security activities increase the cost of the software
	CSR42	Timing attacks and lack of log optimization
	CSR43	Inability to run software updates or change usernames and passwords
	CSR44	Lack of government assistance for proper rules for cybercrime

4.2 Application of Analytic Hierarchy (AHP)

Throughout this part, we determine the relative importance of each investigated software security risk and between each category. Additionally, by completing all of the methods outlined in Section3.4.3, the most critical category of critical software security risks was established.

Step 1: Simplify a complex decision-making problem by dividing it into a hierarchical structure

Shameem et al. [39], Albayrak [79] and Akbar et al. [36] stated that decision-making problems are broken down into a series of interconnected components at this level. Figure 7 shows the problem's hierarchical structure divided into three stages. The problem's aim is mentioned in the first stage of this hierarchical structure; however, the components and sub-factors are situated in stages 2 and3, respectively. The hierarchical structure of the current investigation is depicted in Figure 7.

Step 2: Make a pair-by-pair comparison

This study aims to rank the critical software security risks and their categories in terms of their importance for the successful development of secure software projects. A questionnaire was designed and sent to respondents of the initial survey to conduct the pair-wise comparison (for fuzzy-AHP analysis). Participants in the survey provided a total of 27 replies. To ensure that no data was missing, all replies were rigorously examined. We discovered that all 27 of the responses were complete. Second-survey questionnaire samples are included in Appendix-A. One potential concern with fuzzy-AHP analysis is a small sample size. However, a similar-sized dataset has been utilized in previous publications [80-83] to do the AHP analysis:

Several existing studies also consider FAHP data from a small sample size. For example, Shameem et al. [39] gathered data from 5 specialists to compare the impacting elements of distributed agile development pair-wise. Similarly, Cheng and Li [82] have collected nine responses to a pair-wise comparison of the success variables for building partnering. Furthermore, Wong and Li [80] found FAHP is an effective tool for narrowing down the options for intelligent building systems during a survey of nine industry professionals. Based on these illustrations, we may conclude that a sample size of 27 is sufficient for FAHP.

The survey data from the FAHP participants have been transformed into geometric means for comparing security risks and categories side by side. A geometric mean is an efficient method for

converting survey respondents' assessments into TFN values. The following formula has been used to calculate the geometric mean: The following formula has been used to calculate the geometric mean:

$$\text{Geometric mean} = \sqrt[n]{a1 \times a2 \times a3 \ldots \ldots . an}$$

a = Weight of each response

n = Number of responses

The linguistic variable against their triangular fuzzy Likert scales is given in Table 1. The triangular fuzzy conversion scale (Table 1) suggested by Bozbura et al. [84] was used to create the pair-wise comparison matrixes of the reported critical software security risks and their associated categories.

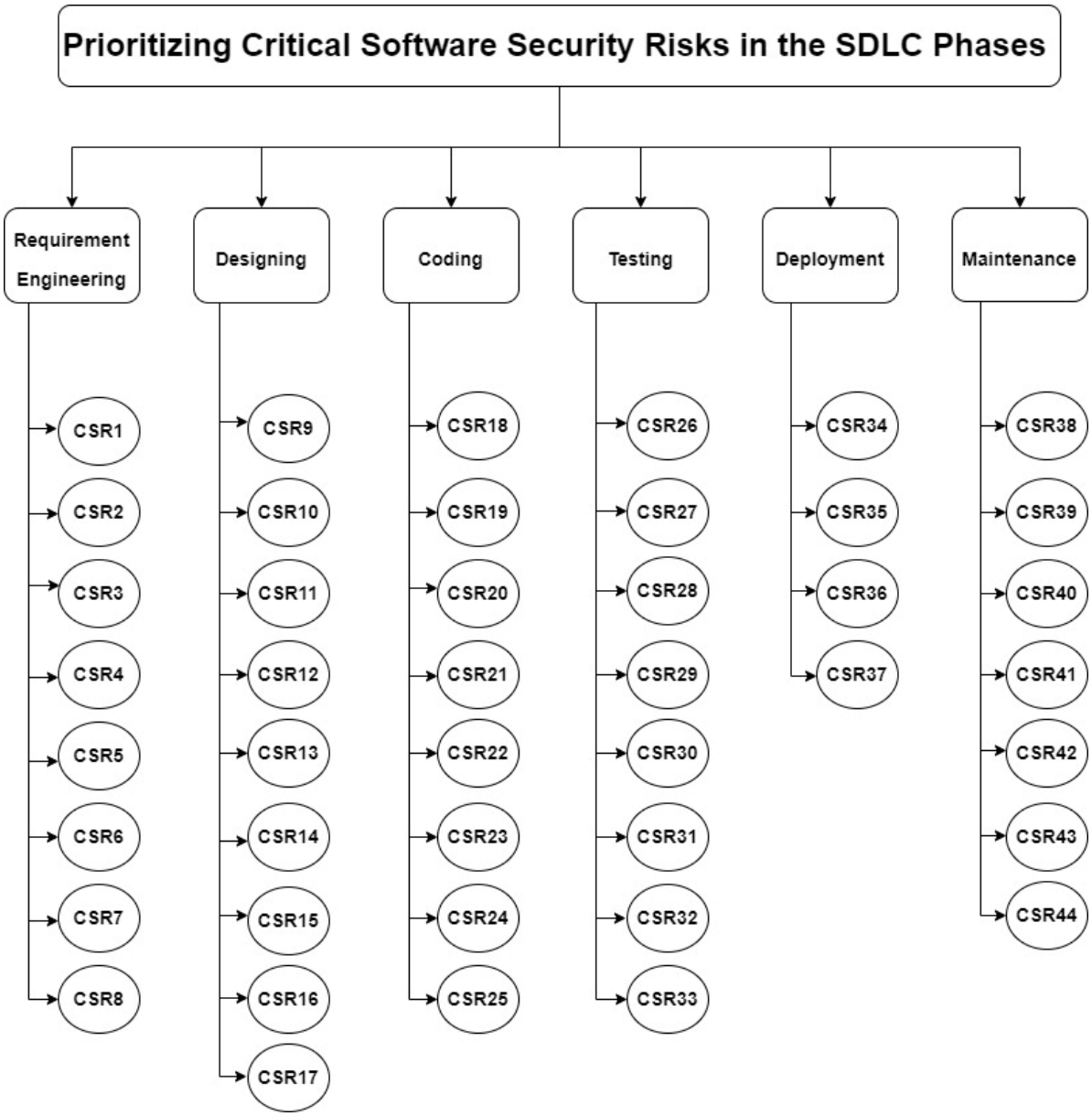

Figure 7: Proposed hierarchical structure of the critical software security risks of SDLC in GSD

Step 3: Determine the local priority weight for each security risk

We calculated the priority weight of each critical security risk (CSR) and their related phases to identify the criticality of each CSR for developing secure software applications in the context of GSD. In the first place, the synthetic extent values of four CSRs in the category "Deployment" were found. As a result, we calculated the priority weight for each CSRs in the following manner. For example, we determined local weights for the CSRs of the "Deployment" category. Table 4 shows a pair-wise analysis of CSRs in the "Deployment" category.

$$\sum_{i}^{n}\sum_{j}^{m} F_{gi}^{j} = (1,1,1) + (1.5,2,2.5) + (1,1.5,2)\ldots+(0.5,0.6,1) + (1,1,1) = (14.1,18.2,22.8)$$

$$\left[\sum_{i}^{n}\sum_{j}^{m} F_{gi}^{j}\right]^{-1} = \left(\frac{1}{22.8},\frac{1}{18.2},\frac{1}{14.1}\right) = (0.04386, 0.054945, 0.070922)$$

$$\sum_{j=1}^{m} F_{g1}^{j} = (1,1,1) + (1.5,2.5,3) + (1,1.5,2) + (1.5,2.0,2.5) = (5,7,8.5)$$

$$\sum_{j=1}^{m} F_{g2}^{j} = (0.3,0.4,0.6) + (1,1,1) + (0.4,0.5,0.6) + (0.5,0.6,1) = (2.2,2.5,3.2)$$

$$\sum_{j=1}^{m} F_{g3}^{j} = (0.5,0.6,1) + (1.5,2,2.5) + (1,1,1) + (1,1.5,2) = (4,5.1,6.5)$$

$$\sum_{j=1}^{m} F_{g4}^{j} = (0.4,0.5,0.6) + (1,1.5,2) + (0.5,0.6,1) + (1,1,1) = (2.9,3.6,4.6)$$

The following equation (4) was used to determine the synthesis values for the "Deployment" category CSRs (CSR34–CSR37):

$$CSR34 = \sum_{j}^{m} F_{g1}^{j} \otimes \left[\sum_{i}^{n}\sum_{j}^{m} F_{gi}^{j}\right]^{-1}$$

$$= (5,7,8) \otimes (0.04376, 0.054955, 0.070922) = (0.219398, 0.384515, 0.601837)$$

$$CSR35 = (2.3,2.5,3.2) \otimes (0.04286, 0.054845, 0.070912) = (0.095491, 0.137463, 0.226940)$$

$$CSR36 = (4,5.1,6) \otimes (0.04376, 0.054845, 0.070822) = (0.174439, 0.281220, 0.450993)$$

$$CSR37 = (2.9,3.6,5.6) \otimes (0.04286, 0.055945, 0.071922) = (0.127183, 0.197812, 0.325241)$$

Table 4: Pair-wise analysis of CSRs in the "Deployment" category

Critical Software Security Risks	CSR34	CSR35	CSR36	CSR37
CSR34	(1,1,1)	(1.5, 1.5, 1)	(1, 1.5, 3)	(1.5, 2, 0.5)
CSR35	(0.5, 1.5, 0.5)	(1,1,1)	(1.5, 0.5, 1.5)	(0.5, 0.5,2)
CSR36	(1, 1.5, 0.6)	(2, 1, 2.5)	(1,1,1)	(2.5, 1, 1.5)
CSR37	(0.5, 2.5, 0.5)	(1.5, 1.5, 1.5)	(1.5, 0.5, 2)	(1,1,1)

Table 5: Results of V values for criteria (priority weight)

	CSR34	CSR35	CSR36	CSR37	d (Priority Weight)
V (R34≥….)	-	1	1	1	1
V (R35≥….)	0.0301	-	0.2670	0.6256	0.0303

V (R36≥….)	0.6983	1	-	1	0.6972
V (R37≥….)	0.3643	1	0.6467	-	0.3661

Table 6: Fuzzy Crisp Matrix (FCM) critical software security risks in the deployment phase

	CSR34	**CSR35**	**CSR36**	**CSR37**	**Priority Weight**
CSR34	0.5	2.5	0.5	1.5	0.11571
CSR35	1.5	1	1.5	1.5	0.29510
CSR36	0.7	2.0	1.0	1.0	0.17128
CSR37	1.5	1.5	0.7	2.5	0.41892

Equation 6 was used to compute the degree of possibility, and equation 8 was used to get the minimal degree of possibility (priority weights) for each pair-wise comparison. Table 5 shows the estimated weights, which are W'= W (1, 0.0303, 0.6972, 0.3661).The statistical significance of the data was determined by normalizing it to W =(0.085234, 0.098666, 0.085853, 0.088267, 0.073985, 0.083276).

Step 4: Consistency Check

We conducted the consistency check using Table 6 and followed all the steps necessary to determine the consistency of pair-wise comparison matrixes. As shown in Table 6, the Fuzzy Crisp Matrix (FCM) was created by converting the fuzzy triangular numbers of the pair-wise comparison matrix of the "Deployment" category CSRs into crisp numbers. To find the greatest Eigen vector (max), we added the sums of the columns of the FCM matrix together. We then divided each value by the total of its appropriate column to find the greatest Eigen vector (max) (Table 6). Finally, In Table 7, we take the average of each row and divide it by the total number of CSRs to assess their priority weight.

Table 7: Normalized matrix of CSRs in the deployment phase

	CSR34	**CSR35**	**CSR36**	**CSR37**
CSR34	0.3703	0.3563	0.4065	0.3835
CSR35	0.1853	0.1438	0.1343	0.1354
CSR36	0.2593	0.2848	0.2714	0.2876
CSR37	0.1855	0.2154	0.1889	0.1932

$$\lambda_{max} = \Sigma\ ([\Sigma\ Cj] \times \{W) \qquad (17)$$

ΣCj= present the sum of Matrix [C] columns (Table 7),

W= present the weight (Table 7), therefore

λ_{max} = 2.7*0.3835+ 7.0*0.1354 + 3.7*0.2876+ 5.2*0.1932= 4.0520

The λ_{max} of FCM matrix is 4.0520 and the matrix size is 4×4, thus according to the RI value given in Table-2 is 0.9.

Furthermore, we used the Equation 15 and Equation 16, CR is determined as follows:

$$CI = \frac{\lambda_{max} - n}{n-1} = \frac{4.0520-4}{4-1} = \frac{0.0520}{3} = 0.0173 \qquad (15)$$

$$CR = \frac{CI}{RI} = \frac{0.0173}{0.9} = 0.0192 \qquad (16)$$

A consistent pair-wise comparison of the "Deployment" category was found to have a calculated value of CR of 0.0192< 0.10, which indicates that the category is consistent. Similarly, the consistency ratio for all the categories is evaluated, and the results, together with a pair-wise comparison of "Requirement Engineering", "Designing", "Coding", "Testing", and "Maintenance", are reported in Tables 8, 9, 10 and 11.

Table 8: Pair-wise comparison and priority weight of CSRs in the requirement engineering phase

	CSR1	CSR2	CSR3	CSR4	CSR5	CSR6	CSR7	CSR8	Priority Weight
CSR1	(1,1,1)	(1, 1.5, 2)	(0.5, 1, 0.5)	(1, 0.5, 2.5)	(0.5, 0.5, 1.5)	(2.5, 2, 0.5)	(1, 1.5, 0.5)	(1.5, 0.5, 1)	0.099521
CSR2	(1.5, 0.6, 1)	(1,1,1)	(1.5, 2, 2)	(0.5, 0.5, 1)	(0.5, 0.5, 1)	(1, 0.5, 2.5)	(1.5, 2, 1)	(0.5, 0.5, 1)	0.095747
CSR3	(1.5, 2, 1.5)	(0.5, 1.5, 1.5)	(1,1,1)	(0.5, 1, 0.5)	(1.5, 0.5, 1)	(1.5, 1, 1.5)	(0.5, 0.5, 1)	(1.5, 1, 2.5)	0.089021
CSR4	(1.5, 0.5, 1)	(1, 1.5, 1.5)	(0.5, 1, 1.5)	(1,1,1)	(1, 1.5, 2)	(0.5, 0.5, 1)	(0.5, 0.5, 1)	(1, 0.5, 2)	0.094227
CSR5	(1, 2.5, 2)	(1.5, 2, 2)	(1, 1.5, 0.5)	(1.5, 0.5, 1)	(1,1,1)	(1, 1.5, 0.5)	(1, 0.5, 2.5)	(1.5, 0.5, 0.5)	0.106190
CSR6	(1.5, 0.5, 1.5)	(1.5, 0.5, 1)	(0.6, 1, 0.5)	(1.5, 0.5, 2)	(1, 2.5, 1.5)	(1,1,1)	(0.5, 1, 0.6)	(1, 0.5, 2.5)	0.073994
CSR7	(1.5, 0.5, 1)	(1.5, 0.5, 1)	(1.5, 1, 2)	(1, 0.5, 2)	(1.5, 0.5, 1)	(1, 2, 1.5)	(1,1,1)	(01.5, 0.5, 1)	0.085222
CSR8	(1, 1.5, 2.5)	(1.5, 1, 2.5)	(0.5, 1, 2.5)	(0.5, 0.5, 1)	(0.5, 1.5, 0.5)	(0.5, 0.5, 1)	(1.5, 1, 1.5)	(1,1,1)	0.098655

Table 9: Pair-wise comparison and priority weight of CSRs in the designing phase

	CSR9	CSR10	CSR11	CSR12	CSR13	CSR14	CSR15	CSR16	CSR17	Priority Weight
CSR9	(1,1,1)	(1, 1.5, 1)	(0.5, 1, 0.5)	(2.5, 2, 0.5)	(1.5, 0.5, 1)	(1.5, 1, 1.5)	(1.5, 1, 2.5)	(1.5, 0.5, 1)	(1.5, 1, 2)	0.089761
CSR10	(1, 0.5, 1.5)	(1,1,1)	(1.5, 2, 1)	(0.5, 1, 1.5)	(0.5, 0.5, 1.5)	(1.5, 1, 2)	(1.5, 2, 2.5)	(0.5, 0.5, 1)	(1, 1.5, 2)	0.092647
CSR11	(1, 1.5, 0.5)	(0.5, 0.5, 1)	(1,1,1)	(0.5, 1.5, 0.5)	(1.5, 0.5, 1)	(1.5, 2, 1.5)	(0.5, 1, 0.5)	(1.5, 1, 0.5)	(0.5, 1.5, 1)	0.082365
CSR12	(0.5, 0.5, 1)	(1, 1.5, 0.5)	(1.5, 0.5, 1)	(1,1,1)	(0.5, 2, 1.5)	(0.5, 1.5, 1)	(1.5, 0.5, 1)	(1.5, 1, 0.5)	(1.5, 0.5, 1)	0.074265
CSR13	(1, 0.5, 1.5)	(1.5, 2, 1.5)	(1, 1.5, 0.5)	(0.5, 1.5, 0.5)	(1,1,1)	(0.5, 0.5, 1)	(0.5, 1, 1.5)	(1, 2, 2.5)	(1.5, 0.5, 1)	0.099316
CSR14	(1, 0.5, 1.5)	(1.5, 0.5, 1)	(0.5, 1.5, 0.5)	(1, 0.5, 1.5)	(1, 1.5, 1.5)	(1,1,1)	(0.5, 1, 1.5)	(1.5, 0.5, 2)	(1.5, 0.5, 1)	0.072521
CSR15	(1.5, 0.5, 1)	(0.5, 1.5, 1)	(1.5, 1, 2.5)	(1, 1.5, 2)	(1.5, 2, 2.5)	(1, 1.5, 2)	(1,1,1)	(0.4, 0.5, 0.6)	(0.5, 0.6, 1)	0.083386
CSR16	(1.5, 0.5, 2)	(1.5, 0.5, 1)	(0.5, 1.5, 1)	(0.5, 0.5, 2)	(1.5, 0.5, 0.5)	(1.5, 0.5, 1)	(1.5, 2, 2.5)	(1,1,1)	(1.5, 2, 2.5)	0.093576
CSR17	(2.5, 0.5, 0.5)	(2.5, 0.5, 1)	(1, 0.5, 2)	(1.5, 0.5, 2)	(1, 1.5, 2)	(1.5, 2, 1.5)	(1, 1.5, 2)	(0.5, 0.5, 1.5)	(1,1,1)	0.074578

Table 10: Pair-wise comparison and priority weight of CSRs in the coding phase

	CSR18	CSR19	CSR20	CSR21	CSR22	CSR23	CSR24	CSR25	Priority Weight
CSR18	(1,1,1)	(1, 0.5, 1.5)	(1.5, 0.5, 0.5)	(1, 1.5, 2)	(0.5, 1.5, 1)	(1, 0.5, 2)	(1, 1.5, 2)	(1.5, 0.5, 1)	0.110114
CSR19	(1.5, 0.1, 1)	(1,1,1)	(1.5, 2, 1.5)	(1.5, 0.5, 1)	(0.5, 1, 1.5)	(1, 0.5, 2)	(1.5, 2, 2.5)	(0.5, 1.5, 1)	0.098378
CSR20	(1.5, 2, 1.5)	(0.5, 0.5, 1.5)	(1,1,1)	(1.5, 0.5, 1)	(0.5, 1.1, 1)	(1, 1.5, 2)	(0.5, 1, 0.5)	(1.5, 0.5, 2.5)	0.095145
CSR21	(1.5, 0.5, 1)	(1, 1.5, 0.5)	(0.5, 1.5, 1)	(1,1,1)	(1.5, 2, 2.5)	(0.5, 1.5, 1)	(0.5, 1, 0.5)	(1.5, 0.5, 1.5)	0.089665
CSR22	(1.5, 0.5, 2)	(0.5, 2, 1.5)	(1, 0.5, 2)	(1.5, 0.5, 0.5)	(1,1,1)	(1.5, 0.5, 0.5)	(1.5, 0.5, 1.5)	(1.5, 2, 1.5)	0.109771
CSR23	(1.5, 0.5, 1)	(1.5, 0.5, 1)	(1.5, 0.5, 1)	(1.5, 0.5, 2)	(1, 0.5, 2)	(1,1,1)	(1.5, 0.5, 0.5)	(1, 1.5, 2)	0.087083
CSR24	(1.5, 0.5, 1)	(0.5, 1.5, 0.5)	(1.5, 2, 1.5)	(1, 0.5, 2.5)	(1.5, 2, 2.5)	(1.5, 2, 1.5)	(1,1,1)	(0.5, 0.5, 1)	0.119217
CSR25	(1, 1.5, 0.5)	(1, 0.5, 2)	(0.5, 1.5, 0.6)	(0.5, 1.5, 0.5)	(0.5, 1.5, 0.5)	(1.5, 0.5, 1)	(1, 1.5, 2)	(1,1,1)	0.099248

Table 11: Pair-wise comparison and priority weight of CSRs in the testing phase

	CSR26	CSR27	CSR28	CSR29	CSR30	CSR31	CSR32	CSR33	Priority Weight

CSR26	(1,1,1)	(0.5, 0.6, 1)	(0.4, 0.5, 0.6)	(1.5, 2, 2.5)	(0.5, 1.5, 1)	(0.5, 1.5, 0.5)	(1.5, 2, 2.5)	(0.5, 2, 2.5)	0.078233
CSR27	(1, 1.5, 2)	(1,1,1)	(1.5, 2, 2.5)	(0.5, 0.6, 1)	(1.5, 0.5, 0.5)	(1.5, 2, 0.5)	(1.5, 0.5, 1)	(1.5, 2, 0.5)	0.077154
CSR28	(1, 1.5, 2)	(0.4, 0.5, 0.6)	(1,1,1)	(0.4, 0.5, 0.6)	(1, 1.5, 2)	(1.5, 1.5, 0.5)	(1, 1.5, 2)	(1.5, 0.5, 2)	0.061136
CSR29	(0.4, 0.5, 0.6)	(1, 1.5, 2)	(1.5, 2, 2.5)	(1,1,1)	(1, 1.5, 2)	(0.5, 1.5, 0.5)	(1.5, 2, 2.5)	(0.5, 1.5, 1)	0.072115
CSR30	(1.5, 2, 2.5)	(1.5, 2, 2.5)	(0.4, 0.5, 0.6)	(0.5, 0.6, 1)	(1,1,1)	(0.5, 1, 2.5)	(1.5, 1.5, 1)	(1, 1.5, 0.5)	0.075752
CSR31	(1.5, 2, 2.5)	(0.4, 0.5, 0.6)	(1.5, 2, 2.5)	(1.5, 2, 2.5)	(1.5, 0.5, 1)	(1,1,1)	(1, 1.5, 2)	(0.5, 1.5, 1)	0.074992
CSR32	(0.4, 0.5, 0.6)	(1, 1.5, 2)	(0.4, 0.5, 0.6)	(0.4, 0.5, 0.6)	(1, 1.5, 2)	(1.5, 1, 1.5)	(1,1,1)	(1, 1.5, 2)	0.065515
CSR33	(0.4, 0.5, 0.6)	(0.4, 0.5, 0.6)	(2.5, 3, 3.5)	(1, 1.5, 2)	(1.5, 1, 2.5)	(1.5, 1, 2.5)	(0.5, 1.5, 1)	(1,1,1)	0.077155

Table 12: Pair-wise comparison and priority weight of CSRs in the maintenance phase

	CSR38	**CSR39**	**CSR40**	**CSR41**	**CSR42**	**CSR43**	**CSR44**	**Priority Weight**
CSR38	(1,1,1)	(1.5, 0.5, 1)	(1.5, 2, 2.5)	(1.5, 2, 2.5)	(0.5, 1.5, 1)	(0.5, 0.5, 1)	(1.5, 2, 2.5)	0.106170
CSR39	(1, 0.5, 2)	(1,1,1)	(1.5, 0.5, 2)	(1, 1.5, 2)	(0.5, 0.5, 0.6)	(1, 1.5, 2)	(0.5, 1.5, 1)	0.073974
CSR40	(0.5, 1.5, 1)	(1.5, 2, 2.5)	(1,1,1)	(0.4, 0.5, 0.6)	(0.5, 1.5, 1)	(1.5, 2, 2.5)	(0.5, 0.5, 1)	0.085242
CSR41	(1.5, 0.5, 1)	(0.5, 1.5, 1)	(1.5, 2, 2.5)	(1,1,1)	(1.5, 2, 2.5)	(1, 1.5, 2)	(1, 1.5, 2)	0.098675
CSR42	(1, 0.5, 2)	(1.5, 2, 2.5)	(1, 1.5, 2)	(0.5, 1.5, 0.6)	(1,1,1)	(0.5, 0.5, 0.5)	(0.5, 1.5, 1)	0.085862
CSR43	(1, 1.5, 2)	(0.5, 1.5, 1)	(1, 0.5, 1.5)	(0.5, 1.5, 1)	(1.5, 2, 2.5)	(1,1,1)	(1.5, 2, 2.5)	0.088267
CSR44	(1, 0.5, 1.5)	(1, 1.5, 2)	(1, 1.5, 2)	(1.5, 2, 1)	(1, 2.5, 2)	(1.5, 0.5, 0.5)	(1,1,1)	0.083375

Table 13: Pair-wise comparison and priority weight among SDLC phases in GSD

SDLC Phases	**Deployment**	**Requirement Engineering**	**Designing**	**Coding**	**Testing**	**Maintenance**	**Priority Weight**
Deployment	(1,1,1)	(0.4, 0.5, 0.6)	(1, 1.5, 2)	(0.4, 0.5, 0.6)	(1, 1.5, 2)	(0.5, 0.6, 1)	0.073985
Requirement Engineering	(1.5, 2, 2.5)	(1,1,1)	(0.4, 0.5, 0.6)	(0.5, 0.6, 1)	(1.5, 2, 2.5)	(0.5, 0.6, 1)	0.085234
Designing	(0.5, 0.6, 1)	(1.5, 2, 2.5)	(1,1,1)	(1.5, 2, 2.5)	(1, 1.5, 2)	(1, 1.5, 2)	0.098666
Coding	(1.5, 2, 2.5)	(1, 1.5, 2)	(0.4, 0.5, 0.6)	(1,1,1)	(0.4, 0.5, 0.6)	(0.5, 0.6, 1)	0.085853
Testing	(0.5, 0.6, 1)	(0.4, 0.5, 0.6)	(0.5, 0.6, 1)	(1.5, 2, 2.5)	(1,1,1)	(1.5, 2, 2.5)	0.088267
Maintenance	(1, 1.5, 2)	(1, 1.5, 2)	(0.5, 0.6, 1)	(1, 1.5, 2)	(0.4, 0.5, 0.6)	(1,1,1)	0.083276

Table 14: Pair-wise comparison and priority weight among SDLC phases and its critical software security risks

Category	**Category Weight**	**Risk**	**local Weight**	**Local Rank**	**Global Weight**	**Global Rank**
Requirement Engineering	0.085234	CSR1	0.099521	2	0.0084826	15
		CSR2	0.095747	4	0.0081609	21
		CSR3	0.089021	6	0.0075876	25
		CSR4	0.094227	5	0.0080313	23
		CSR5	0.106190	1	0.009051	10
		CSR6	0.073994	8	0.0063068	40
		CSR7	0.085222	7	0.0072638	29
		CSR8	0.098655	3	0.0084088	17
Designing	0.098666	CSR9	0.089761	4	0.0088564	11
		CSR10	0.092647	3	0.0091411	9
		CSR11	0.082365	6	0.0081266	22
		CSR12	0.074265	8	0.0073274	30
		CSR13	0.099316	1	0.0097991	5
		CSR14	0.072521	9	0.0071554	31
		CSR15	0.083386	5	0.0082274	18
		CSR16	0.093576	2	0.0092328	8
		CSR17	0.074578	7	0.0073583	27
Coding		CSR18	0.110114	2	0.0094536	6

	0.085853	CSR19	0.098378	5	0.008446	16
		CSR20	0.095145	6	0.0081685	20
		CSR21	0.089665	7	0.007698	24
		CSR22	0.109771	3	0.0094242	7
		CSR23	0.087083	8	0.0074763	26
		CSR24	**0.119217**	**1**	**0.0102351**	**4**
		CSR25	0.099248	4	0.0085207	14
Testing	0.088267	CSR26	0.078233	1	0.0069054	35
		CSR27	0.077154	3	0.0068102	36
		CSR28	0.061136	8	0.0053963	43
		CSR29	0.072115	6	0.0063654	39
		CSR30	0.075752	4	0.0066864	37
		CSR31	0.074992	5	0.0066193	38
		CSR32	0.065515	7	0.0057828	42
		CSR33	0.077155	2	0.0068102	36
Deployment	0.073985	CSR34	0.11571	4	0.0085608	13
		CSR35	**0.29510**	**2**	**0.021833**	**2**
		CSR36	**0.17128**	**3**	**0.0126722**	**3**
		CSR37	**0.41892**	**1**	**0.0309938**	**1**
Maintenance	0.083276	CSR38	0.106170	1	0.0088414	12
		CSR39	0.073974	7	0.0061603	41
		CSR40	0.085242	5	0.0070986	33
		CSR41	0.098675	2	0.0082173	19
		CSR42	0.085862	4	0.0071502	32
		CSR43	0.088267	3	0.0073505	28
		CSR44	0.083375	6	0.0069431	34

Step-5: Determine the relative importance of critical software security risks on a local and global weight

Table 14 shows the local and global weights for each critical software security risk (CSR) and its SDLC phase. When a CSR is given a local weight, it indicates its importance to other similar CSRs. On the other hand, the global weight reveals which of the 44CSRs has the highest priority. The pair-wise comparison was used to generate each CSR and category (step4).

For instance, Table 14 indicates that CSR5, "Improper plan for secure requirement identification, inception, authentication, authorization, and privacy", has the highest local weight (LW) in the Requirement Engineering category (0.106190), indicating that CSR5is the highest ranked (prioritized) component in the Requirement Engineering category.

However, each CSR's global weight was computed by multiplying its local weight by the weight of its category. For example, the global weight (GW) of CSR1 = weight of its category (Requirement Engineering) multiply with it local weight = 0.085234 × 0.099521 = 0.0084826 as depicted in Table 14. Similarly, we estimated each reported CSR's global weight (GW) (see Table 14).

The results in Table 14 indicate that CSR37 (Lack of threat models updating, GW=0.0309938) is the highest-ranking critical software security risk in the SDLC process implementation in the GSD context.

Step 6: Prioritizing of CSRs

The final stage of the AHP process prioritizes the critical software security risks facing software development organizations. The application of this prioritizing is that software development organizations will focus on these priority base CSRs during the SDLC phases to develop a secure software application/product. This prioritization of CSRs is depicted in Table 15. The global weights are used to determine the final rankings of CSRs and calculate their relative importance.

However, the absolute rankings in Table 15 show that CSR37 (Lack of threat models updating, GW=0.0309938) is the most important CSR. This indicates that the secure software development experts have acknowledged the need for effective security management to help software development organizations effectively perform the GSD activities. We further noted (see Table 15) that CSR35 (Lack

of output validation), CSR36 (Lack of certification in the final release and archive), CSR24 (Spoofing: An attempt to gain access to a system by using a fake identity), are the second, third and fourth highest-ranking CSRs in the SDLC process in the GSD context.

Table 15: List of CSRs and their priority order/global rank

Code / Label	CSRs in the SDLC phases	Priority order / global rank
CSR37	Lack of threat models updating	1
CSR35	Lack of output validation	2
CSR36	Lack of certification in final release and archive	3
CSR24	Spoofing: An attempt to gain access to a system by using a fake identity	4
CSR13	Lack of attention to follow security design principles	5
CSR18	Tampering: is the unauthorized modification of data	6
CSR22	Information Disclosure: is the unwanted exposure of private data	7
CSR16	Lack of security design awareness, guidance, and training	8
CSR10	Lack of developing threat modeling during the design phase	9
CSR5	Improper plan for secure requirement identification, inception, authentication, authorization, and privacy	10
CSR9	Improper secure design documentation and specification review	11
CSR38	Lack of proper methods to find out new threats in the system	12
CSR34	Lack of default software configuration	13
CSR25	Password Conjecture: Lack of password complexity enforcement	14
CSR1	Lack of security requirements, review, assessment, analysis, verification, validation	15
CSR19	SQL Injection, Cross Site Scripting, cross-site request forgery	16
CSR8	Lack of security requirements prioritization, management, and categorization	17
CSR15	Lack of building and maintaining abuse case models and attack patterns	18
CSR41	Security activities increase the cost of the software	19
CSR20	Denial of Services: is the process of making a system or application unavailable	20
CSR2	Security requirements are often neglected or considered a non-functional requirement	21
CSR11	Lack of access control and traceability	22
CSR4	Lack of experience, knowledge, guidance, and security training during security requirement documentation	23
CSR21	Repudiation: is the ability of users (legitimate or otherwise) to deny that they performed specific actions or transactions	24
CSR3	Lack of secure requirements identification and documentation	25
CSR23	Elevation of privilege: occurs when a user with limited privileges assumes the identity of a privileged user	26
CSR17	Lack of implementation of security design decisions: (Cryptographic protocols, standards, services, frameworks, abuse case models, and attack patterns)	27
CSR43	Inability to run software updates or change usernames and passwords	28
CSR7	Lack of developing threat modeling	29
CSR12	Improper security design review and its verification	30
CSR14	Lack of developing data flow diagrams and design requirements	31
CSR42	Timing attacks and lack of log optimization	32
CSR40	Improper configuration, vulnerability management, change control, and improvement of security assessment	33
CSR44	Lack of government assistance for proper rules for cybercrime	34
CSR26	Lack of Penetration Security Testing Analysis	35
CSR27	Lack of Static and Dynamic Security Testing Analysis	36
CSR33	Various kinds of Attacks (viruses,) malware, Trojan Virus: A type of virus that is well known for causing issues and destruction to computers is a Trojan virus	36
CSR30	Brute Force Attack	37

CSR31	Lack of developing threat models: as it helps to build test cases or test plans	38
CSR29	Lack of Fuzz and Unit Testing Analysis	39
CSR6	Lack of security requirements elicitation activity	40
CSR39	Lack of security trust	41
CSR32	Lack of Functional and Non-Functional Testing	42
CSR28	Lack of final and manual security review	43

5. Results Evaluation and Analysis

The fuzzy analytical hierarchy approach (FAHP) is used in this paper to prioritize the critical software security risks (CSRs) in the SDLC phases in the global software development (GSD) domain. As stated in Section1, we have generated three research questions to accomplish the study's goal. The results of the evaluation and the analysis are briefly shown in the following sections:

5.1 RQ1: What are the most cited software security risks in the SDLC phases in the domain of GSD?

This study found 44 CSRs (see Table 15) that are essential for the SDLC process in GSD to be carried out successfully. It is based on our prior systematic literature review analysis that identified 145 software security risks [2]. Section 4.1 and Figure 6 describe the mapping approach we used to group the researched CSRs into six essential categories based on the framework developed by Shameem et al. [39]. The primary objective of security risks categorization is to establish a hierarchy process that will aid in applying the fuzzy AHP. It is also important for practitioners and academic researchers to consider the CSRs for SDLC phases implementation and future study, respectively, by using the mapping technique.

The most cited/critical software security risk is CSR37 "Lack of threat models updating", CSR35 "Lack of output validation", CSR36 "Lack of certification in final release and archive", CSR24 "Spoofing: An attempt to gain access to a system by using a fake identity" in the SDLC process in the GSD context.

5.2 **RQ2: What is the best way to prioritize the significant software security risks?**

The fuzzy AHP approach prioritized the examined CSRs and their categories. This was accomplished using pair-wise comparisons between the security risks and the relevant categories. The pair-wise comparison is important in determining the importance of the CSRs in the SDLC process in the context of GSD. Every CSR and its category were ranked following the priorities set. The fuzzy AHP technique enables a thorough comprehension of multi-criteria decision-making situations that incorporate the significance of SDLC process security improvement and their associated categories.

It is clear from Table 15 that CSR37, "Lack of threat models updating", is the most important CSR in the SDLC phases. For this reason, secure SDLC experts believe the software development organization should have qualified and skilled team members. On the other hand, the members of secure software development organizations must be capable of efficient communication and coordination [85, 86].

5.3 RQ3: What would be the decision-making framework for the critical software security risks?

Figure 8: Decision-Making Framework

Finally, taking into consideration the framework proposed by Shameem et al. [39], we develop a taxonomy by classifying the prioritized CSRs into six main categories (Secure Requirement Engineering (RE), Secure Designing, Secure Coding, Secure Testing and Review, Secure Deployment, and Secure Maintenance) in Figure 8. Each CSR is given a global and local weight in the taxonomy based on its importance. There are two types of weights: local and global. The local weight shows how a CSR affects its category, and the global weight affects the whole SDLC process. As a result, the taxonomy (Figure 8) aids in specifying the influence of a single CSR in a specific category and across the entire SDLC process.

Figure 8 shows that the most CSRs are: CSR37 "Lack of threat models updating (LW = 0.41892 and GW = 0.0309938)", CSR35 "Lack of output validation (LW = 0.29510 and GW = 0.021833)", CSR36 "Lack of certification in final release and archive (LW = 0.17128 And GW = 0.0126722)", CSR24 "Spoofing: An attempt to gain access to a system by using a fake identity (LW = 0.119217 and GW = 0.0102351)" in the SDLC process in the GSD context.

Threat modeling is a systematic method for identifying threats that may compromise security, and it is considered a well-known accepted practice by the software testing industry [87]. In CLASP, threat modeling and risk analysis are performed during the requirement and design phase. The design and implementation phase suggests secure design guidelines and coding standards [88]. Microsoft uses STRIDE to model threats to their systems; threats are defined by looking into the possibilities of spoofing identity, tampering with data, repudiation, information leakage, denial of services, and elevation in the given situation [89]. Incorrect input/output validation refers to the lack of or inaccurate validation of input/output provided by a user via the application's user interface. Injection attacks take advantage of the lack of input/output validation controls to allow malicious inputs to be passed in, which can be used to obtain elevated rights, alter data, or crash a system [90]. Code injection attacks can breach data security, cause a loss of services, and harm thousands of users' systems [91]. During the secure deployment phase, final security reviews and audits are performed [11, 36]. At this phase, customer satisfaction is also very important.

Table 14 shows that "Designing" (W = 0.098666) has been declared the most important category of the CSRs evaluated by software development industry experts. Software development experts should pay close attention to the CSRs of the designing category. Moreover, the "Testing" category (W = 0.088267), followed by "Coding" (W = 0.085853), are the second and third most important categories, respectively. Table 14 indicate that category (Designing, category weight = 0.098666) is the highest-ranking category based on software security in the SDLC process implementation. The design phase is one of the most creative stages of the SDLC, which is one of the reasons it is important from the viewpoint of security [4, 92]. 50 % of software defects are identified and detected during the design stage of the SDLC [4, 92, 93]. Design-level flaws are software systems' most common security risks [4]. Designation-level flaws are software systems' most common security risks [4]. Building secure software means building software that functions properly even under malicious attacks [94]. This requires addressing the security challenges throughout the SDLC, especially in the early stages of the design phase [95]. This reduces the risk of overlooking critical security requirements or introducing security flaws throughout the implementation process. To complete this phase appropriately and securely, the software developer must consider security best practices during design.

6. Study Limitations

It is necessary to elaborate potential risks of this study before using the findings in industry or for other research purposes. For example, the sample size of FAHP survey responses is n = 27, which may not be strong enough to support the validity of the reported CSRs when evaluating their explanation. Compared to the previous survey, the FAHP survey had a smaller sample size (27 replies), threatening the

generalizability of our results. The FAHP method, on the other hand, has been viewed as a more subjective approach that allows for smaller sample size [65-68, 96].

Construct validity measures how well an assessment scale is used to examine the supplied CSRs' work. A questionnaire survey was conducted with real-world software development practitioners to determine the significance of the researched elements concerning the SDLC process in GSD. Following the findings, the parameters identified could favour the SDLC process in the GSD context.

Internal validity is the evaluation of the results and analyses presented in the study. The results of pilot research we conducted with secure software development specialists show that the study has an adequate level of internal validity.

External validity refers to the ability of a study's results to be generalized. The survey participants in this study came from various continents and nations, yet we are convinced that the data sample was sufficiently representative and generalizable.

7. Study Implications

It is hoped that the findings of this study will have both practical and research consequences because they give a prioritized set of SDLC critical software security risks, which will serve as a knowledge base for industry practitioners and academic scholars in the field of GSD. The classification of the researched CSRs aids the researchers in determining the most important category of CSR to consider when planning their future studies. Furthermore, the study gives a prioritization-based taxonomy of the aspects that contributed to the success of secure software development applications.

Prioritization-based taxonomy helps GSD practitioners consider the most important CSRs related to their specific categories based on the most important local rankings. Using the taxonomy, the practitioners (software development organizations) may see the global rankings of each CSR, making it easier for them to identify the CSR while implementing secure SDLC-related requirements modifications. In summary, this study presents a complete analysis of the GSD and secure SDLC critical software security risks and their priority order, which has not been done in this field. We believe that the study's findings will aid software development industry practitioners in building effective plans for successfully executing secure SDLC operations in the context of GSD.

8. Conclusion and Future Directions

An increase in GSD projects prompted us to study and prioritize the aspects that could positively impact SDLC. The CSRs that have been documented emphasize the critical areas that must be addressed with urgency for the secure SDLC process to be successfully implemented in the GSD. One hundred forty-five software security risks were found due to the systematic literature review approach that we have used [2]. The security risks observed were then divided into six distinct software development process improvement areas. Furthermore, a questionnaire survey has been used to determine what software development industry professionals think about the security risks of SDLC phases in GSD. According to the empirical data analysis, more than 70% of survey participants agreed that the enlisted security risks are important to consider when developing secure software in the GSD context.

As a result of the FAHP approach, the "Designing" category is by far the most significant identified CSRs. In addition, the FAHP results show that "Lack of threat models updating", "Lack of output validation", and "Lack of certification in final release and archive" are the three most important CSRs in the designing category.

Software development organizations in the GSD domain could benefit from a taxonomy of CSRs that can amend and evaluate their SDLC strategies based on the categorization, weighting, and prioritizing of CSRs. The taxonomy presented here is based on a hierarchical and multidimensional model that incorporates the significance of SDLC activities in the GSD environment in the six core categories.

We believe that the findings, analysis, and conclusions of this study will be useful in addressing the issues related to the improvement of the SDLC process, which is critical to the success and development of GSD businesses.

With the increasing number of software security threats, regularly upgrade software security processes and practices. This study project can be improved in a variety of ways. The following are some of the open study directions that researchers can look into near future:

- We intend to develop a security tool from a Security Assurance Model (SAM) of Software Development [16, 97] for global software development (GSD) vendor organizations. This model will assist GSD vendors in determining their readiness for secure software development. We will develop the model using the results of this study, SLR, industrial survey, case study, supervisor inputs, and lessons learned from the existing studies [5, 8, 12, 98-100]. The model will generate several assessment reports, including a list of security risks/threats and practices that GSD vendor organizations will use in each phase of the SDLC.
- Collaboration with software development organizations is required to improve the outcomes of SAM of Software Development. Depending on the facilities and methods used, it might be adapted to meet the needs of various organizations.
- The SAM of Software Development might include characteristics relating to specific technologies like the Internet of Things (IoT), blockchain, and cloud computing.
- The SAM of Software Development might be made available as an online repository (tool) updated regularly with new academic and industry practices. The SAM of Software Development will become a reliable resource for scholars and practitioners.

Acknowledgment

We thank the Software Engineering Research Group at the University of Malakand, the questionnaire survey, and FAHP Survey participants for their critiques and contributions to this project (SERG UOM).

References

[1] A. Ramirez, A. Aiello, and S. J. Lincke, "A Survey and Comparison of Secure Software Development Standards," in *2020 13th CMI Conference on Cybersecurity and Privacy (CMI) - Digital Transformation - Potentials and Challenges*, Copenhagen, Denmark, pp. 1-6, 2020.

[2] R. A. Khan, S. U. Khan, H. U. Khan, and M. Ilyas, "Systematic Literature Review on Security Risks and its Practices in Secure Software Development," *IEEE Access*, vol. 10, pp. 5456-5481, 2022.

[3] H. Nina, J. A. Pow-Sang, and M. Villavicencio, "Systematic Mapping of the Literature on Secure Software Development," *IEEE Access*, vol. 9, pp. 36852-36867, 2021.

[4] R. A. Khan, S. U. Khan, H. U. Khan, and M. Ilyas, "Systematic Mapping Study on Security Approaches in Secure Software Engineering," *IEEE Access*, vol. 9, pp. 19139-19160, 2021.

[5] J. C. S. Núñez, A. C. Lindo, and P. G. Rodríguez, "A Preventive Secure Software Development Model for a Software Factory: A Case Study," *IEEE Access*, vol. 8, pp. 77653-77665, 2020.

[6] S. V. Solms and L. A. Futcher, "Adaption of a Secure Software Development Methodology for Secure Engineering Design," *IEEE Access*, vol. 8, pp. 125630-125637, 2020.

[7] M. Z. Gunduz and R. Das, "Cyber-security on smart grid: Threats and potential solutions," *Computer Networks*, vol. 169, p. 1-14, 2020.

[8] M. Niazi, A. M. Saeed, M. Alshayeb, S. Mahmood, and S. Zafar, "A maturity model for secure requirements engineering," *Computers & Security*, vol. 95, p. 1-34, 2020.

[9] M. Zhang, X. d. C. d. Carnavalet, L. Wang, and A. Ragab, "Large-Scale Empirical Study of Important Features Indicative of Discovered Vulnerabilities to Assess Application Security," *IEEE Transactions on Information Forensics and Security*, pp. 1-12, 2019.

[10] G. McGraw, "Six Tech Trends Impacting Software Security," *Computer*, vol. 50, pp. 100-102, 2017.
[11] R. Khan, "Secure software development: a prescriptive framework," *Computer Fraud & Security*, vol. 2011, pp. 12-20, 2011.
[12] H. Al-Matouq, S. Mahmood, M. Alshayeb, and M. Niazi, "A Maturity Model for Secure Software Design: A Multivocal Study," *IEEE Access*, vol. 8, pp. 215758-215776, 2020.
[13] S. Moyo and E. Mnkandla, "A Novel Lightweight Solo Software Development Methodology With Optimum Security Practices," *IEEE Access*, vol. 8, pp. 33735-33747, 2020.
[14] R. A. Khan, S. U. Khan, M. Ilyas, and M. Y. Idris, "The State of the Art on Secure Software Engineering: A Systematic Mapping Study," presented at the Proceedings of the Evaluation and Assessment in Software Engineering, Trondheim, Norway, pp. 1-6, 2020.
[15] G. McGraw, "From the ground up: the DIMACS software security workshop," *IEEE Security & Privacy*, vol. 99, pp. 59-66, 2003.
[16] R. A. Khan and S. U. Khan, "A preliminary structure of software security assurance model," presented at the Proceedings of the 13th International Conference on Global Software Engineering, Gothenburg, Sweden, pp. 137-140, 2018.
[17] B. Potter and G. McGraw, "Software security testing," *IEEE Security & Privacy*, vol. 2, pp. 81-85, 2004.
[18] D. Verdon and G. McGraw, "Risk Analysis in Software Design," *IEEE Security and Privacy*, vol. 2, pp. 79–84, 2004.
[19] Sodiya, A.S. Onashoga, S. A., and Ajayi, O. B.: 'Towards Building Secure Software Systems', Issues in Informing Science and Information Technology, Vol 3, 2006, pp. 636-636
[20] I. Flechais, C. Mascolo, and M. A. Sasse, "Integrating security and usability into the requirements and design process," *Int. J. Electron. Secur. Digit. Forensic*, vol. 1, pp. 12–26, 2007.
[21] S. Gupta, M. Faisal, and M. Husain, "Secure Software Development Process for Embedded Systems Control," *International Journal of Engineering Sciences & Emerging Technologies*, vol. 4, pp. 133-143, 2012.
[22] Team, C.P.: 'CMMI for Development, Version 1.3', in Editor (Ed.): 'Book CMMI for Development, Version 1.3' (Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2010, edn.), pp. 1-482.
[23] R. Al-Qutaish and A. Abran, "A Maturity Model of Software Product Quality," *Journal of Research and Practice in Information Technology*, vol. 43, pp. 307-327, 2011.
[24] B. Golden: 'Succeeding with Open Source. Reading, ' in Editor (Ed.): 'Book Succeeding with Open Source. Reading, ' (2005, edn.), pp. 1-242.
[25] A. April, J. Huffman Hayes, A. Abran, and R. Dumke, "Software Maintenance Maturity Model (SMmm): the software maintenance process model," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 17, pp. 197-223, 2005.
[26] O. Turetken, I. Stojanov, and J. Trienekens, "Assessing the adoption level of scaled agile development: a maturity model for Scaled Agile Framework (SAFe)," *Journal of Software: Evolution and Process*, vol. 29, pp. 1-18, 2016.
[27] M. P. d. Silva and R. M. d. Barros, "Maturity Model of Information Security for Software Developers," *IEEE Latin America Transactions*, vol. 15, pp. 1994-1999, 2017.
[28] S. R. Ahmed, Secure Software Development : Identification of Security Activities and Their Integration in Software Development Lifecycle', in Editor (Ed.): 'Book Secure Software Development : Identification of Security Activities and Their Integration in Software Development Lifecycle' (2007, edn.), pp. 1-40.
[29] M. Essafi, L. Jilani, and H. Ben Ghezala: 'S2D-ProM: A Strategy Oriented Process Model for Secure Software Development' 2007, pp. 1-24.
[30] J. Manico, "OWASP " in *Application Security Verification Standard 3.0.1*, ed, 2016, pp. 1-70.
[31] BSIMM: 'Building security in maturity model (BSIMM)', in Editor (Ed.)^(Eds.): 'Book Building security in maturity model (BSIMM)' (2022, edn.), pp. 1-65.

[32] N. Mead and T. Stehney, "Security quality requirements engineering (SQUARE) methodology," *ACM SIGSOFT Software Engineering Notes*, vol. 30, pp. 1-7, 2005.
[33] C.-C. Sun, "A performance evaluation model by integrating fuzzy AHP and fuzzy TOPSIS methods," *Expert Systems with Applications*, vol. 37, pp. 7745-7754, 2010.
[34] M. A. Akbar, K. Smolander, S. Mahmood, A. J. I. Alsanad, and S. Technology, "Toward successful DevSecOps in software development organizations: A decision-making framework," *Information Software Technology*, vol. 147, p. 1-21, 2022.
[35] M. A. Akbar, S. Rafi, A. A. Alsanad, S. F. Qadri, A. Alsanad, and A. J. I. A. Alothaim, "Toward Successful DevOps: A Decision-Making Framework," *IEEE Access*, vol. 10, pp. 51343-51362, 2022.
[36] M. A. Akbar, A. A. Khan, Z. J. I. T. Huang, and Management, "Multicriteria decision making taxonomy of code recommendation system challenges: a fuzzy-AHP analysis," *Information Technology and Management*, pp. 1-17, 2022.
[37] T. Kamal, Q. Zhang, M. A. Akbar, M. Shafiq, A. Gumaei, and A. Alsanad, "Identification and Prioritization of Agile Requirements Change Management Success Factors in the Domain of Global Software Development," *IEEE Access*, vol. 8, pp. 44714-44726, 2020.
[38] T. Saaty, "Analytic hierarchy process," ed, 2001, pp. 19-28.
[39] M. Shameem, R. R. Kumar, C. Kumar, B. Chandra, and A. A. Khan, "Prioritizing challenges of agile process in distributed software development environment using analytic hierarchy process," *Journal of Software: Evolution and Process*, vol. 30, p. 1-19, 2018.
[40] R. d. F. S. M. Russo and R. Camanho, "Criteria in AHP: A Systematic Review of Literature," *Procedia Computer Science*, vol. 55, pp. 1123-1132, 2015.
[41] S. Ali, H. Li, S. U. Khan, Y. Zhao, and L. Li, "Fuzzy Multi Attribute Assessment Model for Software Outsourcing Partnership Formation," *IEEE Access*, vol. 6, pp. 55431-55461, 2018.
[42] M. A. Akbar, A. Alsanad, S. Mahmood, and A. Alothaim, "A Multicriteria Decision Making Taxonomy of IOT Security Challenging Factors," *IEEE Access*, vol. 9, pp. 128841-128861, 2021.
[43] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering – A systematic literature review," *Information and Software Technology*, vol. 51, pp. 7-15, 2009.
[44] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," pp. 1-65. 2007.
[45] N. Dissanayake, A. Jayatilaka, M. Zahedi, and M. A. Babar, "Software security patch management - A systematic literature review of challenges, approaches, tools and practices," *Information and Software Technology*, vol. 144, p. 1-45, 2022.
[46] A. Shukla, B. Katt, L. O. Nweke, P. K. Yeng, and G. K. Weldehawaryat, "System security assurance: A systematic literature review," *Computer Science Review*, vol. 45, p. 1-30, 2022.
[47] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, pp. 1-26, 2004.
[48] W. Afzal, R. Torkar, and R. Feldt, "A systematic review of search-based testing for non-functional system properties," *Information and Software Technology*, vol. 51, pp. 957-976, 2009.
[49] R. Torkar, C. A. Furia, R. Feldt, F. G. d. O. Neto, L. Gren, P. Lenberg, *et al.*, "A Method to Assess and Argue for Practical Significance in Software Engineering," *IEEE Transactions on Software Engineering*, pp. 1-1, 2021.
[50] L. Zhang, J. Tian, and J. Jiang, "Empirical Research in Software Engineering — A Literature Survey," *Jounrla of Computer Science Technology*, vol. 33, pp. 876–899, 2018.
[51] S. Xu, "Empirical research methods for software engineering: Keynote address," in *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)*, 2017, pp. 1-1.
[52] M. De Stefano, E. Iannone, F. Pecorelli, and D. A. Tamburri, "Impacts of software community patterns on process and product: An empirical study," *Science of Computer Programming*, vol. 214, p. 1-20, 2022.

[53] M. Nevendra and P. Singh, "Empirical investigation of hyperparameter optimization for software defect count prediction," *Expert Systems with Applications*, vol. 191, p. 116-216, 2022.
[54] S. Wagner, D. M. Fernández, M. Felderer, A. Vetrò, M. Kalinowski, R. Wieringa, *et al.*, "Status Quo in Requirements Engineering: A Theory and a Global Family of Surveys," *ACM Trans. Softw. Eng. Methodol.*, vol. 28, p. 1-48, 2019.
[55] M. Niazi, D. Wilson, and D. Zowghi, "Critical success factors for software process improvement implementation: an empirical study," *Software Process: Improvement and Practice*, vol. 11, pp. 193-211, 2006.
[56] H. U. Rahman, M. Raza, P. Afsar, and H. U. Khan, "Empirical Investigation of Influencing Factors Regarding Offshore Outsourcing Decision of Application Maintenance," *IEEE Access*, vol. 9, pp. 58589-58608, 2021.
[57] M. A. Akbar, W. Naveed, A. A. Alsanad, L. Alsuwaidan, A. Alsanad, A. Gumaei, *et al.*, "Requirements Change Management Challenges of Global Software Development: An Empirical Investigation," *IEEE Access*, vol. 8, pp. 203070-203085, 2020.
[58] H. Mumtaz, M. Alshayeb, S. Mahmood, and M. Niazi, "An empirical study to improve software security through the application of code refactoring," *Information and Software Technology*, vol. 96, pp. 112-125, 2018.
[59] A. Mazuera-Rozo, C. Escobar-Velásquez, J. Espitia-Acero, D. Vega-Guzmán, C. Trubiani, M. Linares-Vásquez, *et al.*, "Taxonomy of security weaknesses in Java and Kotlin Android apps," *Journal of Systems and Software*, vol. 187, p. 1-11, 2022.
[60] C. Beaman, M. Redbourne, J. D. Mummery, and S. Hakak, "Fuzzing vulnerability discovery techniques: Survey, challenges and future directions," *Computers & Security*, vol. 120, p. 1-13, 2022.
[61] B. Kitchenham and S. L. Pfleeger, "Principles of survey research part 6: data analysis," *SIGSOFT Softw. Eng. Notes*, vol. 28, pp. 24–27, 2003.
[62] A. A. Khan, J. Keung, M. Niazi, S. Hussain, and A. Ahmad, "Systematic literature review and empirical investigation of barriers to process improvement in global software development: Client–vendor perspective," *Information and Software Technology*, vol. 87, pp. 180-205, 2017.
[63] B. Martin, *Introduction to Medical Statistics*, 4th Edition ed., pp. 1-464, 2015.
[64] S. U. Khan, M. Niazi, and A. Rashid, "Factors influencing clients in the selection of offshore software outsourcing vendors: an exploratory study using a systematic literature review," *Journal of Systems and Software*, vol. 84, pp. 686-699, 2011.
[65] T. Yaghoobi, "Prioritizing key success factors of software projects using fuzzy AHP," *Journal of software: Evolution and process*, vol. 30, p. e1891, 2018.
[66] E. Sloane, M. Liberatore, R. Nydick, W. Luo, and Q. Chung, "Clinical engineering technology assessment decision support: a case study using the analytic hierarchy process (AHP)," in *Proceedings of the Second Joint 24th Annual Conference and the Annual Fall Meeting of the Biomedical Engineering Society][Engineering in Medicine and Biology*, 2002, pp. 1950-1951.
[67] L. Wen-ying, "Application of ahp analysis in risk management of engineering projects [j]," *Journal Beijing University of Chemical Technology (Social Sciences Edition)*, vol. 1, pp. 46-48, 2009.
[68] G. Kabra, A. Ramesh, and K. Arshinder, "Identification and prioritization of coordination barriers in humanitarian supply chain management," *International Journal of Disaster Risk Reduction*, vol. 13, pp. 128-138, 2015.
[69] G. J. Klir, *Fuzzy set theory*: Wiley-IEEE Press, pp. 260-314, 2006.
[70] M. A. Akbar, H. Alsalman, A. A. Khan, S. Mahmood, C. Meshram, A. H. Gumaei, *et al.*, "Multicriteria decision making taxonomy of cloud-based global software development motivators," *IEEE Access*, vol. 8, pp. 185290-185310, 2020.
[71] C.-K. Kwong and H. Bai, "A fuzzy AHP approach to the determination of importance weights of customer requirements in quality function deployment," *Journal of intelligent manufacturing*, vol. 13, pp. 367-377, 2002.

[72] C.-K. Kwong and H. Bai, "Determining the importance weights for the customer requirements in QFD using a fuzzy AHP with an extent analysis approach," *IEEE Transactions*, vol. 35, pp. 619-626, 2003.
[73] M. A. Akbar, M. Shameem, S. Mahmood, A. Alsanad, and A. J. Gumaei, "Prioritization based taxonomy of cloud-based outsource software development challenges: Fuzzy AHP analysis," *IEEE Access*, vol. 95, p. 106557, 2020.
[74] L. A. Zadeh, G. J. Klir, and B. Yuan, *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers* vol. 6: World Scientific, pp. 1-840, 1996.
[75] T. L. Saaty, "Analytic hierarchy process," *Encyclopedia of operations research and management science*, pp. 52-64, 2013.
[76] P. J. Van Laarhoven and W. Pedrycz, "A fuzzy extension of Saaty's priority theory," *Fuzzy sets and Systems*, vol. 11, pp. 229-241, 1983.
[77] D.-Y. Chang, "Applications of the extent analysis method on fuzzy AHP," *European journal of operational research*, vol. 95, pp. 649-655, 1996.
[78] J. M. Corbin and A. Strauss, "Grounded theory research: Procedures, canons, and evaluative criteria," *Qualitative sociology*, vol. 13, pp. 3-21, 1990.
[79] E. Albayrak and Y. C. Erensal, "Using analytic hierarchy process (AHP) to improve human performance: An application of multiple criteria decision making problem," *Journal of Intelligent Manufacturing*, vol. 15, pp. 491-503, 2004.
[80] J. K. Wong and H. Li, "Application of the analytic hierarchy process (AHP) in multi-criteria analysis of the selection of intelligent building systems," *Building and Environment*, vol. 43, pp. 108-125, 2008.
[81] S. Soh, "A decision model for evaluating third-party logistics providers using fuzzy analytic hierarchy process," *African Journal of Business Management*, vol. 4, pp. 339-349, 2010.
[82] E. W. Cheng and H. Li, "Construction partnering process and associated critical success factors: quantitative investigation," *Journal of management in engineering*, vol. 18, pp. 194-202, 2002.
[83] K. Lam and X. Zhao, "An application of quality function deployment to improve the quality of teaching," *International Journal of Quality & Reliability Management*, vol. 15, pp. 389-413, 1998.
[84] F. T. Bozbura, A. Beskese, and C. Kahraman, "Prioritization of human capital measurement indicators using fuzzy AHP," *Expert Systems with Applications*, vol. 32, pp. 1100-1112, 2007.
[85] N. M. Minhas and A. Zulfiqar, "An improved framework for requirement change management in global software development," *Journal of Software Engineering and Applications*, vol. 7, pp. 1-779, 2014.
[86] N. Ali and R. Lai, "A method of requirements elicitation and analysis for Global Software Development," *Journal of Software: Evolution and Process*, vol. 29, pp. 1-27, 2017.
[87] D. Basin, J. Doser, and T. Lodderstedt, "Model driven security: From UML models to access control infrastructures," *ACM Trans. Softw. Eng. Methodol.*, vol. 15, pp. 39–91, 2006.
[88] W. Li and T. Chiueh, "Automated Format String Attack Prevention for Win32/X86 Binaries," in *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, 2007, pp. 398-409.
[89] L. Y. Banowosari and B. A. Gifari, "System Analysis and Design Using Secure Software Development Life Cycle Based On ISO 31000 and STRIDE. Case Study Mutiara Ban Workshop," in *2019 Fourth International Conference on Informatics and Computing (ICIC)*, 2019, pp. 1-6.
[90] M. Almorsy, J. Grundy, and A. S. Ibrahim, "Automated software architecture security risk analysis using formalized signatures," in *2013 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 662-671.
[91] R. Cope, "Strong security starts with software development," *Network Security*, vol. 2020, pp. 6-9, 2020.

[92] V. Maheshwari and M. Prasana, "Integrating Risk assessment and Threat modeling within SDLC process," in *International Conference on Inventive Computation Technologies (ICICT)*, 2016, pp. 1-5.
[93] M. A. Akbar, S. Mahmood, and D. Siemon, "Toward Effective and Efficient DevOps using Blockchain," in *The International Conference on Evaluation and Assessment in Software Engineering*, 2022, pp. 421-427.
[94] N. S. A. Karim, A. Albuolayan, T. Saba, and A. Rehman, "The practice of secure software development in SDLC: an investigation through existing model and a case study," *Security and Communication Networks*, vol. 9, pp. 5333-5345, 2016.
[95] S. Ahmed, "Secure Software Development : Identification of Security Activities and Their Integration in Software Development Lifecycle," pp. 1-40, 2007.
[96] M. A. Akbar, A. A. Khan, S. Mahmood, A. J. I. T. Mishra, and Management, "SRCMIMM: the software requirements change management and implementation maturity model in the domain of global software development industry," pp. 1-25, 2022.
[97] R. A. Khan, S. U. Khan, M. Alzahrani, and M. Ilyas, "Security Assurance Model of Software Development for Global Software Development Vendors," *IEEE Access*, pp. 58458-58488, 2022.
[98] Asad. Muhammad and A. Shafique, "Model Driven Architecture for Secure Software Development Life Cycle " *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 14, pp. 649-661, 2016.
[99] Y. Mufti, M. Niazi, M. Alshayeb, and S. Mahmood, "A Readiness Model for Security Requirements Engineering," *IEEE Access*, vol. 6, pp. 28611-28631, 2018.
[100] R. A. Khan, M. Y. Idris, S. U. Khan, M. Ilyas, S. Ali, A. U. Din, *et al.*, "An Evaluation Framework for Communication and Coordination Processes in Offshore Software Development Outsourcing Relationship: Using Fuzzy Methods," *IEEE Access*, vol. 7, pp. 112879-112906, 2019.

Appendix-A: (Sample of Fuzzy AHP Survey Questionnaire**)**

Section- A1 (Respondent Information)			
Full Name (optional)		Position	
Secure Software Development working experience (Years)?			
Email Address			
Address of your current organization including the country name			
Total academic/industrial experience in years			
Have you ever participated in Secure Software Development process?	Yes		No
Section- A2 (Organization Detail)			

Current organization name (Optional)		
What is the primary business of your organization? (You may tick more than one)	Global/offshore Development ☐	Collocated/single site development ☐
	Research ☐	Other ☐
Size of your current organization	Small ☐	Medium ☐
	Large ☐	Not sure ☐
Please specify your organization type	National ☐	Multinational ☐
	Not sure ☐	Other ☐
Does your organization adopt Secure Software Developmentstandards or models?(Please specify)		
For how long your organization is using the Secure Software Development standard/model? (Years)		

Section B- Pair-wise Comparison of the Identified Critical Software Security Risks (CSRs) and Categories

Categories (SDLC Phases)	**Code / Label**	**Critical Security Risks (CSRs)**
Requirement Engineering	CSR1	"Lack of security requirements, review, assessment, analysis, verification, validation"
	CSR2	"Security requirements are often neglected or considered as a non-functional requirement"
	CSR3	"Lack of secure requirements identification and documentation"
	CSR4	"Lack of experience, knowledge, guidance, and security training during security requirement documentation"
	CSR5	"Improper plan for secure requirement identification, inception, authentication, authorization and privacy"
	CSR6	"Lack of security requirements elicitation activity"
	CSR7	"Lack of developing threat modeling"
	CSR8	"Lack of security requirements prioritization, management and

		categorization"
Designing	CSR9	"Improper secure design documentation and specification review"
	CSR10	"Lack of developing threat modeling during the design phase"
	CSR11	"Lack of access control and traceability"
	CSR12	"Improper security design review and its verification"
	CSR13	"Lack of attention to follow security design principles"
	CSR14	"Lack of developing data flow diagrams and design requirements"
	CSR15	"Lack of building and maintaining abuse case models and attack patterns"
	CSR16	"Lack of security design awareness, guidance, and training"
	CSR17	"Lack of implementation of security design decisions: (Cryptographic protocols, standards, services, frameworks, abuse case models and attack patterns)"
Coding	CSR18	"Tampering: is the unauthorized modification of data"
	CSR19	"SQL Injection, Cross Site Scripting, cross-site request forgery"
	CSR20	"Denial of Services: is the process of making a system or application unavailable"
	CSR21	"Repudiation: is the ability of users (legitimate or otherwise) to deny that they performed specific actions or transactions"
	CSR22	"Information Disclosure: is the unwanted exposure of private data"
	CSR23	"Elevation of privilege: occurs when a user with limited privileges assumes the identity of a privileged user"
	CSR24	"Spoofing: An attempt to gain access to a system by using a fake identity"
	CSR25	"Password Conjecture: Lack of password complexity enforcement"
Testing	CSR26	"Lack of Penetration Security Testing Analysis"
	CSR27	"Lack of Static and Dynamic Security Testing Analysis"
	CSR28	"Lack of final and manual security review"
	CSR29	"Lack of Fuzz and Unit Testing Analysis"
	CSR30	"Brute Force Attack"
	CSR31	"Lack of developing threat models: as it helps to develop test cases or test plans"
	CSR32	"Lack of Functional and Non Functional Testing"

	CSR33	"Various kinds of Attacks (viruses,) malware, Trojan Virus: A type of virus that is well known for causing issues and destruction to computers is a Trojan virus"
Deployment	CSR34	"Lack of default software configuration"
	CSR35	"Lack of output validation"
	CSR36	"Lack of certification in final release and archive"
	CSR37	"Lack of threat models updating"
Maintenance	CSR38	"Lack of proper methods to find out new threats in the system"
	CSR39	"Lack of security trust"
	CSR40	"Improper configuration, vulnerability management, change control and improvement of security assessment"
	CSR41	"Security activities increase the cost of the software"
	CSR42	"Timing attacks and lack of log optimization"
	CSR43	"Inability to run software updates or change usernames and passwords"
	CSR44	"Lack of government assistance for proper rules for cybercrime"

Perform the pair-wise comparison of the CSRs and the given categories by putting the checkmark (✓). For example, if a CSR on the left side of Table 1 is more significant than the right side, then put the checkmark on the left side of the scale just equal (JE) based on your preference. Similarly, if a category on the right side of Table1 is more important than the matching category at the left, then put the checkmark on the right side of the scale just equal (JE).

Table 1 refersto a questionnaire hierarchal structure to determine the weight priorities of the CSRs and their respective categories by putting the checkmark on the pair-wise comparison matrices. For example, how important is CSR1 as compared to CSR2 when we execute SDLCimprovement activities in global software development domain.

Description	Significance intensity
Just equal (JE)	(1,1,1)
Equally important (EI)	(1/2,1,3/2)
Weakly important (WI)	(1,3/2,2)
Strong more important (SMI)	(3/2,2,5/2)
Very strong more important (VSMI)	(2,5/2,3)

To perform the pair-wise comparison of the CSRS and the given categories by putting the checkmark (✓). For example, if a CSR on the left side of Table 1 is more significant than the right side, then put the checkmark on the left side of the scale i.e. 1 (Equal) based on your preference. Similarly, if a category on the right side of Table1 is more important than the matching category at the left, then put the checkmark on the right side of the scale 1 (Equal).
Table 1 refersto a questionnaire hierarchal structure to determine the weight priorities of the SDLCcritical software security risks and their respective categories by putting the checkmark on the pair-wise comparison matrices. For example, how important is CSR1 as compared to CSR2 when we execute SDLC process activities in GSD?

Section C-Pair-wise Assessment

Requirement Engineering												
Alternatives	AMI	VSMI	SMI	WI	EI	JE	EI	WI	SMI	VSMI	AMI	Alternatives
CSR1												CSR1
CSR1												CSR2
CSR1												CSR3
CSR1												CSR4
CSR1												CSR5
CSR1												CSR6
CSR1												CSR7
CSR1												CSR8
CSR2												CSR2
CSR2												CSR3
CSR2												CSR4
CSR2												CSR5
CSR2												CSR6
CSR2												CSR7
CSR2												CSR8
CSR3												CSR3
CSR3												CSR4
CSR3												CSR5
CSR3												CSR6
CSR3												CSR7
CSR3												CSR8
CSR4												CSR4
CSR4												CSR5
CSR4												CSR6
CSR4												CSR7
CSR4												CSR8
CSR5												CSR5
CSR5												CSR6
CSR5												CSR7
CSR5												CSR8
CSR6												CSR6
CSR6												CSR7
CSR6												CSR8
CSR7												CSR7
CSR7												CSR8
CSR8												CSR8
Designing												
Alternatives	AMI	VSMI	SMI	WI	EI	JE	EI	WI	SMI	VSMI	AMI	Alternatives
CSR9												CSR9
CSR9												CSR10
CSR9												CSR11
CSR9												CSR12

CSR9												CSR13
CSR9												CSR14
CSR9												CSR15
CSR9												CSR16
CSR9												CSR17
CSR10												CSR10
CSR10												CSR11
CSR10												CSR12
CSR10												CSR13
CSR10												CSR14
CSR10												CSR15
CSR10												CSR16
CSR10												CSR17
CSR11												CSR11
CSR11												CSR12
CSR11												CSR13
CSR11												CSR14
CSR11												CSR15
CSR11												CSR16
CSR11												CSR17
CSR12												CSR12
CSR12												CSR13
CSR12												CSR14
CSR12												CSR15
CSR12												CSR16
CSR12												CSR17
CSR13												CSR13
CSR13												CSR14
CSR13												CSR15
CSR13												CSR16
CSR13												CSR17
CSR14												CSR14
CSR14												CSR15
CSR14												CSR16
CSR14												CSR17
CSR15												CSR15
CSR15												CSR16
CSR15												CSR17
CSR16												CSR16
CSR16												CSR17
CSR17												CSR17
Coding												
Alternatives	AMI	VSMI	SMI	WI	EI	JE	EI	WI	SMI	VSMI	AMI	Alternatives
CSR18												CSR18
CSR18												CSR19
CSR18												CSR20
CSR18												CSR21
CSR18												CSR22
CSR18												CSR23
CSR18												CSR24
CSR18												CSR25
CSR19												CSR19
CSR19												CSR20

CSR19												CSR21
CSR19												CSR22
CSR19												CSR23
CSR19												CSR24
CSR19												CSR25
CSR20												CSR20
CSR20												CSR21
CSR20												CSR22
CSR20												CSR23
CSR20												CSR24
CSR20												CSR25
CSR21												CSR21
CSR21												CSR22
CSR21												CSR23
CSR21												CSR24
CSR21												CSR25
CSR22												CSR22
CSR22												CSR23
CSR22												CSR24
CSR22												CSR25
CSR23												CSR23
CSR23												CSR24
CSR23												CSR25
CSR24												CSR24
CSR24												CSR25
CSR25												CSR25
Testing												
Alternatives	AMI	VSMI	SMI	WI	EI	JE	EI	WI	SMI	VSMI	AMI	Alternatives
CSR26												CSR26
CSR26												CSR27
CSR26												CSR28
CSR26												CSR29
CSR26												CSR30
CSR26												CSR31
CSR26												CSR32
CSR26												CSR33
CSR27												CSR27
CSR27												CSR28
CSR27												CSR29
CSR27												CSR30
CSR27												CSR31
CSR27												CSR32
CSR27												CSR33
CSR28												CSR28
CSR28												CSR29
CSR28												CSR30
CSR28												CSR31
CSR28												CSR32
CSR28												CSR33
CSR29												CSR29
CSR29												CSR30
CSR29												CSR31
CSR29												CSR32

CSR29												CSR33
CSR30												CSR30
CSR30												CSR31
CSR30												CSR32
CSR30												CSR33
CSR31												CSR31
CSR31												CSR32
CSR31												CSR33
CSR32												CSR32
CSR32												CSR33
CSR33												CSR33
Deployment												
Alternatives	AMI	VSMI	SMI	WI	EI	JE	EI	WI	SMI	VSMI	AMI	Alternatives
CSR34												CSR34
CSR34												CSR35
CSR34												CSR36
CSR34												CSR37
CSR35												CSR35
CSR35												CSR36
CSR35												CSR37
CSR36												CSR36
CSR36												CSR37
CSR37												CSR37
Maintenance												
Alternatives	AMI	VSMI	SMI	WI	EI	JE	EI	WI	SMI	VSMI	AMI	Alternatives
CSR38												CSR38
CSR38												CSR39
CSR38												CSR40
CSR38												CSR41
CSR38												CSR42
CSR38												CSR43
CSR38												CSR44
CSR38												CSR39
CSR38												CSR40
CSR38												CSR41
CSR38												CSR42
CSR38												CSR43
CSR38												CSR44
CSR39												CSR39
CSR39												CSR40
CSR39												CSR41
CSR39												CSR42
CSR39												CSR43
CSR39												CSR44
CSR40												CSR40
CSR40												CSR41
CSR40												CSR42
CSR40												CSR43
CSR40												CSR44
CSR41												CSR41
CSR41												CSR42
CSR41												CSR43
CSR41												CSR44

CSR42												CSR42
CSR42												CSR43
CSR42												CSR44
CSR43												CSR43
CSR43												CSR44
CSR44												CSR44
Pair-wise Assessment of the Categories (SDLC Phases)												
Alternatives	AMI	VSMI	SMI	WI	EI	JE	EI	WI	SMI	VSMI	AMI	Alternatives
Requirement Engineering												Requirement Engineering
Requirement Engineering												Designing
Requirement Engineering												Coding
Requirement Engineering												Testing
Requirement Engineering												Deployment
Requirement Engineering												Maintenance
Designing												Designing
Designing												Coding
Designing												Testing
Designing												Deployment
Designing												Maintenance
Coding												Coding
Coding												Testing
Coding												Deployment
Coding												Maintenance
Testing												Testing
Testing												Deployment
Testing												Maintenance
Deployment												Deployment
Deployment												Maintenance
Maintenance												Maintenance