Received January 2, 2021, accepted January 14, 2021, date of publication January 18, 2021, date of current version February 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3052311

Systematic Mapping Study on Security Approaches in Secure Software Engineering

RAFIQ AHMAD KHAN[1], SIFFAT ULLAH KHAN[1], HABIB ULLAH KHAN[2], AND MUHAMMAD ILYAS[1]

[1]Software Engineering Research Group, Department of Computer Science and IT, University of Malakand, Chakdara 18800, Pakistan
[2]Department of Accounting and Information Systems, College of Business and Economics, Qatar University, Doha, Qatar

Corresponding author: Habib Ullah Khan (habib.khan@qu.edu.qa)

This work was supported in part by the Qatar National Library, Doha, Qatar, and in part by the Qatar University under Grant IRCC-2020-009.

ABSTRACT In the modern digital era, software systems are extensively adapted and have become an integral component of human society. Such wide use of software systems consists of large and more critical data that inevitably needs to be secured. It is imperative to make sure that these software systems not only satisfy the users' needs or functional requirements, but it is equally important to make sure the security of these software systems. However, recent research shows that many software development methods do not explicitly include software security measures during software development as they move from demand engineering to their final losses. Integrating software security at each stage of the software development life cycle (SDLC) has become an urgent need. Tackling software security, various methods, techniques, and models have been suggested and developed, however, only a few of them provide strong evidence for building secure software applications. The main purpose of this research is to study security measures in the context of the development of secure software (SSD) during the study of systematic mapping (SMS). Based on the inclusion and exclusion criteria, 116 studies were selected. After the data extraction from the selected 116 papers, these were classified based on the quality assessment, software security method, SDLC phases, publication venue, and SWOT analysis. The results indicate that this domain is still immature and sufficient research work needs to be carried out particularly on empirically evaluated solutions.

INDEX TERMS Software security, secure software development, secure software engineering, software development life cycle, security approaches, systematic mapping study.

I. INTRODUCTION

Over the last two decades, the software industry observed phenomenal growth, and the same is continued at a rapid pace. Software is now an important aspect of our lives and it seems almost impossible to find a field that does not have the use of software in their day to day business. The world in every aspect has been modernized by an immense use of software systems.

On the contrary, misuse of software can lead to heavy economic loss in the financial sector, sabotage in the communication sector, critical data theft in databases, and misuse of software in the missile controlling system can endanger human life. Rapid developments in information and communication technologies (ICTs) have made software security a key concern, such as the Internet of Things (IOT) and the Internet of Every Things, the advancement of Internet-based software systems, cloud computing, social networking, and location-based services. Therefore, software programs grow in size, complexity, inclination, and connectivity.

In addressing both the technological and human aspects involved, there is a growing need for understanding Secure Software Engineering (SSE) methods. SSE is about building software that can deal with potentially aggressive attacks, maintaining basic security features: privacy, integrity, and access to sensitive assets [1]–[3]. Also besides, new business paradigms, versatile customers' requirements, rapid advancement in ICTs, and new regulations are constantly making a software application to evolve accordingly [4]. SSE recommends that software security is an important factor to be taken into account during the start of the software life cycle (SDLC) [5]. To build and deploy a secure software

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

system, we need to integrate security features into our life cycle of application development and align current SSE methods [6], [7].

Most businesses view security as a post-development process [8]. Security isn't considered at some point in the pre-development phase [9]. There is no approval for the method to be used, we still have little understanding of the need for secure software development. There are also few facts about the effectiveness of existing approaches to dealing with real problems and a limited view of how they contribute to the assessment of safety concerns [10]. Threats put systems at greater risk for major losses that can be difficult to recover [11]. The majority of software programs are designed and deployed without attention to protection desires [12], [13]. Hidden attacking risks within or outside the organization are emerging day-by-day, results in huge financial loss, as well as confidentiality and credibility losses by putting the availability and integrity of organizational data at risk [14], [15]. The coding phase of SDLC is more prone to error, as the programmer leaves some errors unintentionally, which increases software vulnerability to more attacks [11], [16]. Such vulnerabilities can be the denial of services, code execution, memory corruption/data loss, cross-site scripting (XSS), improper access control, SQL injection, integer overflow, buffer overflow, and the format string, etc., [12], [17].

Based on the above evidence, we can conclude that protecting software programs in the development stages isn't sufficient and there is a great need to locate higher approaches and ways to protect software programs. This paper provides a systematic study of maps (SMS) based on our pre-defined process and the proposed model [18], [19] to highlight the existing security measures for building secure software. It is reported that the findings, in our study reported in this paper, will have an impact on the body of information by providing a tax on the SSE-related research topic that may invite researchers to focus on further research in this field.

We investigate the outcome of this mapping study to provide a reference direction for interested and vigilant researchers to address and explore new research trends and gaps in the subject domain. The findings of the SMS may assist the practitioners by providing a deep insight into the subject domain about security approaches, security limitations, and unresolved specific and general issues.

To achieve the aforementioned objectives, this SMS addresses the following research questions (RQs):

RQ1. What is the state-of-the-art in Secure Software Engineering?

To answer RQ1, we have analyzed the literature based on the following sub-questions:

RQ1.1: What are the existing security methods, implemented by software development organizations that return the largest competitive edge?

RQ1.2: Which particular SDLC phase has been most discussed and addressed in the literature?

RQ1.3: What is the SWOT analysis of security approaches in secure software development?

RQ1.4: What are the popular venues for secure software development?

The rest of the paper is organized as follows: Section II presents the concepts of software security, section III presents the relevant work, and section IV describes the research methodology. The outcomes of the SMS are discussed in section V according to the research questions. Section VI represents the implications for results and practice. Section VII gives the findings and future work. Finally, in section VIII, the risks to validity are addressed.

II. SOFTWARE SECURITY

This section introduces the concept of software security for future discussion that will serve as context building.

A. THE CONCEPTS OF SOFTWARE SECURITY

Let's take a look at some of the software security concepts as stated in the literature:

- "The idea of engineering software that continues to function correctly under malicious attack [7], [20]".
- "The process of designing, building, and testing software for security [21], [22]".
- "Software security is the process of discussing an application to discover risks and vulnerabilities of the application and its data [23]".
- "Software security is a system-wide issue that takes into account both insecurity mechanisms (such as access control) and design for security (such as a robust design that makes software attack difficult)" [24].
- "Software security is about building secure software: designing software to be secure, making sure that software is secure, and educating software developers, architects, and users about to build secure things [25]".
- "Defends against software exploit by building software to be secure in the first place mostly be getting by the design right (which is hard) and avoiding common mistakes (which is easy) [26]".
- "Software security is the ability of the software to resist, tolerate and recover from events that intentionally threaten its dependability [27]".

Security in various terms has been described by various researchers. In the above concepts, the important thing to remember is that most of the definitions speak about "building secure software" instead of "securing software". Building secure software involves designing and securely implementing the software while securing software tries to develop the software first and then enforce security measures to make it secure.

B. SOFTWARE SECURITY REQUIREMENTS

Software security has remained a neglected area, from the earliest generations of software development. But that does not mean that the problem has never been raised before; however, it was misunderstood, taken lightly, misjudged, and not done as it should have been. During the entire software development life cycle, software security is an essential factor that needs to be addressed [18]. In general, security is characterized as a non-functional requirement, and, for this reason, security checks are usually carried out during the final of SDLC [9], [28], [29]. It means that software security requires proper care even in the first stage of software development [9], [15], [28].

Today, Internet-enabled applications, the removal of bugs in the form of buffer overload, and incompatible error management are major issues in software security [22], [30]. Millions of people do business by different means every day, such as the Internet, ATM, cell phone, email, etc. The software is used by people who remember that it is reliable and trustworthy and that the services they perform are secure. But if this includes security gaps, then how can they be considered secure? Security in software has become an important part of daily life. Due to budget constraints and software release time in the market, many developers consider security as a subsequent thinking problem that may have poor software quality [9]. Software security was considered part of software testing in the early days, but over time, it has been shown that security is not a backward concern and it is very important to consider how software engineers can incorporate security into the early stage of SDLC [28].

To this end, this study aims to analyze the process of software development from the perspective of each SDLC phase and to attempt to define key security measures to be used to make the most secure applications in all stages of the SDLC.

C. QUALITIES OF A SECURE SOFTWARE

Secure software is about building software that can withstand strong attacks, maintaining basic security structures: confidentiality, integrity, and access to sensitive assets [31]. These three security structures are called the CIA [32]. Any software that enlists the CIA can be considered as secure software [32]. Software security characteristics are defined as ''the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization'' [33]. Features of secure software are confidentiality, integrity, denial, accountability, and authenticity. Other aspects of software security are as follows [33]:

- Secure Data Transport
- Protect Database Storage
- Authorized Data Access
- Secure Authorization
- Internal / output authentication
- Power of Evidence
- Complete Access
- Alternative Identification
- Accessibility Management capabilities
- Session Management Powers
- Secure User Management

III. BACKGROUND

Have you ever tried to find the cause of insecure software? Who is responsible for unsafe software? Service analysts do their best to find operational and non-operational requirements that meet customer needs. System designers are doing their best to find the most powerful design. Developers are doing their best to improve the app in a very efficient way. Testers do their best to detect software crashes. From demand inquiries to software development, from software testing to maintenance, the entire project team has put all its efforts into the SDLC to ensure quality software but instead of all these efforts, the software is still produced with risks and numerous security errors.

In addition to the extensive efforts made to build secure software and everyone is playing their part to the best of their ability; makes you wonder where these security errors came from? To answer this query, we need to suppose carefully, have we pointed out security to this point? Is security taken into consideration at any level of SDLC? I'm afraid the solution is nearly a large 'NO' at all. Engineers only broaden particular requirements (overall performance necessities) and ignore safety requirements. But it will be more beneficial if they take the security requirements in the earlier phases and following the same in the requirement gathering phases; such as design, development, and testing. And once they incorporate the security at all stages of SDLC, the software developed by adapting the security measures will produce more secure software.

Security activities during the requirement phase serve three purposes [13], [23]: Initial security requirements are identified and implemented. Second, with the security requirements in hand, the project team understands and recognizes the importance of security. Finally, with the needs of security in the hands, budget, resources, and time of security activities in future stages can be better estimated.

During the design phase, the project team focuses on identifying the attacker's interests, potential access points, and critical security areas, etc. [34], [35]. The next step is to identify the threats running on the software. All the security data collected in the design phase so far goes into the threatening model. Threatening models can be considered an important milestone in terms of secure software [36]. Does the security building function provide full details of how the software can be attacked? What can be attacked? What areas of attack are attractive? What kind of threats work etc. [37]. According to this information, the security structure is continuously updated to include security.

The implementation phase plays a twofold role from a security perspective [38], [39]: First, it avoids security errors entering the software, and secondly, detects existing software errors. The first role is done by writing a secure code. The second role of detecting security errors begins with static analysis by automated tools. After automatic analysis, a manual update is performed. After that, the software is fully functional and ready to go to the testing phase.

After implementation, the software is sent to the testing team. The tests were performed mainly on test cases generated during test planning [40]. The testing team identifies security errors, reports to the development team, and the development team corrects them in this code [40], [41]. The testing phase ends when all test cases are conducted, and retrospective testing of all sensitive areas has taken place [42].

Like any other form of testing, security testing involves determining who should do it and what activities they should undertake. Because security testing involves two approaches, the question of who should do it has two answers. Standard testing organizations using a traditional approach can perform functional security testing [43]. For example, ensuring that access control mechanisms work as advertised is a classic functional testing exercise. On the other hand, traditional QA staff will have more difficulty performing risk-based security testing. The problem is one of expertise. First, security tests (especially those resulting in complete exploits) are difficult to craft because the designer must think like an attacker [43]. Second, security tests don't often cause direct security exploits and thus present an observability problem. A security test could result in an unanticipated outcome that requires the tester to perform further sophisticated analysis. Bottom line: risk-based security testing relies more on expertise and experience than we would like [43].

Before the release of the software, a security review was performed [44]. The purpose of the review is to identify the remaining security errors. The developing team corrects code against security errors identified in the review report. After a review, a security audit was conducted, and according to such an audit report, management decided to issue a software [45]. After such a release, the software is ready for shipment.

After release and distribution, the software is commercially used. Later, a decision was made to rectify non-critical safety errors [46]. So another code is changed to remove these security errors in the form of a patch. The patch is then applied to the software after testing and the patch is released [47].

Because of this, to address the software system security various models, practices, strategies, and methods have been proposed and developed to improve security procedures in the stages of SDLC [9], [24], [32], [48]. To effectively address security issues that exist during the application process, it is necessary to consider secure considerations in all development processes that minimize the threats of critical security requirements or to identify critical errors in software development [24].

Some security approaches aim to assist the software engineers in evaluating security risks; such as Attack Trees [49], combining goal-orientation and use-case modeling, which is an effective method of software requirement engineering [50], Secure Tropos, a security-oriented extension to the goal-driven requirements engineering methodology [51], whereas others allow the software engineers to address these risks by reusing design decisions [52] or sustaining the decision making process [36]. Other software security approaches are McGraw's Secure Software Development Life Cycle (SSDLC) process [22], Microsoft Software Development Life Cycle (SDL) or Trustworthy Computing Security Development Life Cycle [53], Security Requirements Engineering Process (SREP) [54], Aprville and Pourzandi's Secure Software Development Life Cycle process [55], Core security requirements artifacts [56], Comprehensive, Lightweight Application Security Process (CLASP) [57], Haley and his colleagues' framework [58], and Security Quality Requirements Engineering (SQUARE) [59].

OWASP Security Verification Standard (ASVS) version 3.0 is a community effort to establish a framework of security requirements and controls that focus on normalizing the functional and non-functional security controls required when designing, developing, and testing modern web applications [60]. The ASVS is a list of application security requirements or tests that can be used by architects, developers, testers, security professionals, and even consumers to define what a secure application is [60].

ISO/IEC 27001:2005 covers all types of organizations (e.g. commercial enterprises, government agencies, not-for-profit organizations) [61]. It specifies the requirements for establishing, implementing, operating, monitoring, reviewing, maintaining, and improving a documented Information Security Management System within the context of the organization's overall business risks. It specifies requirements for the implementation of security controls customized to the needs of individual organizations or parts thereof. It is designed to ensure the selection of adequate and proportionate security controls that protect information assets and give confidence to interested parties.

Browser identity indicators, including URLs and EV certificates, are supposed to help users identify phishing, social engineering, and other attacks, but prior lab studies and surveys suggested that older browser identity UIs are not effective security tools [62]. Modern browser identity indicators are not effective. To design better identity indicators, we recommend that browsers consider focusing on active negative indicators, explore using prominent UI as an opportunity for user education, and incorporate user research into the design phase [62].

We conclude that the most relevant contributions with such goals are the work carried out by Nabil *et al.* [24], Silva *et al.* [63], and Guiena *et al.* [64]. The authors in [24], identified and classified the available software security approaches in SDLC. To identify and mitigate the software security threats, Silva *et al.* [63], covered the current technologies. For the ubiquitous system, Guiena *et al.* [64] identified 132 approaches, address issues in various phases of the software engineering cycle. Most of the studies addressed maintenance/evolution, implementation, and feedback phases.

The main focus of our study is to cover the most relevant SSE models, frameworks, methods, processes, metrics, and topics in the existing literature. The result can be used as a reference guide and direction for future research.

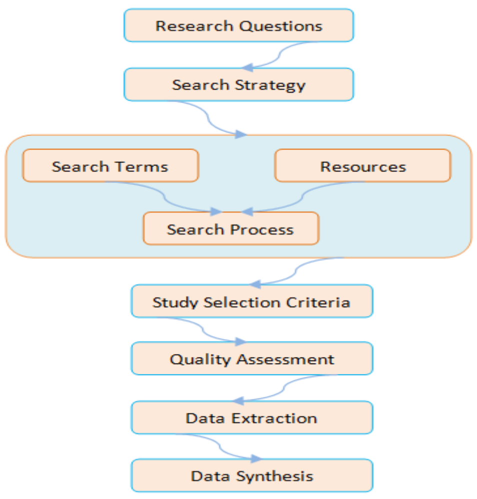


FIGURE 1. **Stages of SMS process.**

IV. RESEARCH METHODOLOGY

The main purpose of the SMS is to provide formal ways to integrate the information found in simple basic lessons with a set of map questions [65]. SMS is a literature study that focuses on selecting and combining all high-quality research, related to a specific topic and provides a complete summary of current texts applicable to specific map queries [66]. Compared to a systematic literature review (SLR), SMS were performed on a wide range of research questions to identify gaps in a particular research area. It also identifies gaps in existing subjects and determines trends in future studies [67]–[69]. SMS, therefore, maintains a great imminent value in the field of software engineering by providing a general view of the literature in this particular domain.

According to Felderer and Carver [70], the method of conducting SMS consists of three main stages: planning, conducting, and reporting. The goal of these phases is to classify, analyze and interpret, based on the strength of their evidence, all available studies relevant to a specific subject, to draw conclusions, and finally provide recommendations [65], [68]–[70]. The various stages used in performing SMS are shown in Figure 1.

A. RESEARCH QUESTIONS

In Table 1, the research questions (RQs) are discussed along with their key motivations.

B. SEARCH STRATEGY

A search using a search string composed of keywords specific to this study was carried out to address the RQs and applied to many academic electronic libraries and search engines.

1) SEARCH STRING

The first step is to establish the search string, and the parameters for PICO (Population, Intervention, Comparison, and Outcomes) have been reported in the literature [65], [71].

C. RESEARCH QUESTIONS

In Table 1, the research questions (RQs) are discussed along with their key motivations.

D. SEARCH STRATEGY

A search using a search string composed of keywords specific to this study was carried out to address the RQs and applied to many academic electronic libraries and search engines.

1) SEARCH STRING

The first step is to establish the search string, and the parameters for PICO (Population, Intervention, Comparison, and Outcomes) have been reported in the literature [65], [71].

- Population: Secure software development
- Intervention: Software Security
- Comparison: Due to the exploratory study, the Comparison is not relevant and so excluded.
  Outcomes: Security Approaches

The search string was constructed on connecting the features of PICO by Boolean AND and OR connectors:

((“software security” OR “software privacy” OR “secure software” OR “software protection” OR “software safety”) AND (“Software Engineering” OR “Software Development lifecycle” OR “SDLC” OR “Software security Model”))

2) LITERATURE RESOURCES

We run the search string in various online digital libraries such as IEEE Xplore, ACM, Springer Link, Science Direct, and Wiley Online Library. In Google Scholar, we also run this search string. Table 2 displays the number of search results per search engine/database.

E. STUDY SELECTION CRITERIA

The search strings were inspired by similar researches [24], [34], [69], [72]–[75] and also from the authors’ suggestions. Initially, the search string was added to the metadata of the selected libraries without affecting the title, abstract, and keyword constraints. The first author retrieved each paper and detailed information was documented about each relevant paper. The other authors were asked to review the articles using their title and abstract, to base the previous steps.

Based on SMS guidelines, the following inclusion and exclusion criteria have been established [65].

1) INCLUSION CRITERIA

- IC1: Papers in the area of software security.
- IC2: Papers related to SDLC.
- IC3: The paper focuses on one or several security approaches from SSD perspectives.

2) EXCLUSION CRITERIA

- EC1: Papers that are not published in journals, conferences, workshops, or symposiums.
- EC2: Papers that aren’t available in English.
- EC3: A research that is not in the Software Engineering domain.

TABLE 1. Research questions and main motivation.

ID	Mapping Questions	Motivations
RQ1	What is the state-of-the-art in Secure Software Engineering?	The main focus of this question is to cover the most up-to-date research in Secure Software Engineering. The result can be used as a reference guide and direction for future research. To answer RQ1, we have analyzed the literature based on the following sub-questions:
RQ1.1.	What are the security methods, implemented by software development organizations that return the largest competitive edge?	To explore the existing methods in the literature related to security in SSD.
RQ1.2	Which particular SDLC phase has been extensively discussed and addressed in the literature?	To explore which particular SDLC phase has been covered and addressed.
RQ1.3	What is the SWOT analysis of security approaches in secure software development?	To make an analysis that explores the strengths, weaknesses, opportunities, and threats of security approaches to SSD reported in the literature.
RQ1.4	What are the popular venues for secure software development?	To explore where the relevant research of the topic can be found and targets for the publications of future studies.

TABLE 2. Search string results per database.

Search String	Digital Libraries	Total Results	Initial Selection	Final Selection
((“software security” OR “software privacy” OR “secure software” OR “software protection” OR “software defense” OR “software safety”) AND (“Software Engineering” OR “Software Development lifecycle” OR “SDLC” OR “Software security Model”))	IEEE Xplore	1,759	177	34
	Science Direct	599	62	31
	ACM	375	79	15
	Springer Link	1,656	52	7
	Wiley Online Library	369	55	9
Google Scholar (Search Engine)		2,570	131	20
Total		7,328	556	116

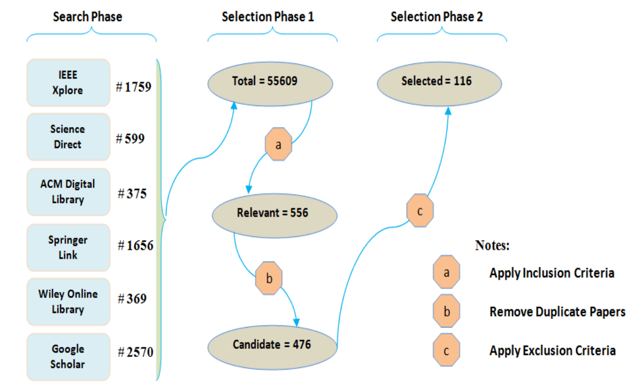


FIGURE 2. Search and selection process.

- EC4: Papers with workshop summaries.
- EC5: The studies which occur several times in the final set (duplicated studies)
- EC6: Books, web pages, and magazine articles are excluded.

The summary of the search process is shown in Figure 2. In total, 556 papers relating to security approaches for SSD were found after the implementation of the inclusion criteria. After the removal of duplicate papers from various sources of the same paper, the remaining papers were 476. Finally, upon applying the exclusion criteria through which we further removed 360 studies, we selected 116 papers as net studies. The paper IDs and titles of the 116 papers finally selected are listed in Appendix-A.

F. QUALITY ASSESSMENT

Quality assessment is the main activity that increases the strength of a study. To enhance the quality of our work, a short questionnaire was therefore designed to assess the quality of candidate papers.

Senior researchers at the Software Engineering Research Group, University of Malakand (SERG_UOM), in Pakistan, distribute and fill out the questionnaire. The scoring plan applied in Appendix A (a, b, c, and d columns) is presented below:

a) The paper was published in a recognized and stable journal or conference. This question was ranked by considering the computer science conference ranking in Computing Research Education (CORE)*, 2018, and Journal Citation Reports (JCR)**, 2018 lists.

Conferences, workshops, and symposiums were assigned to one of the following categories:

- A[1]- Flagship conference, a leading venue in a discipline area. “Very relevant (+2)”
- A - Excellent conference and highly respected one in a discipline area. “Relevant (+1.5)”
- B - Good conference and well regarded one in a discipline area. “Somewhat Relevant (+1)”
- C - Other ranked conferences that meet minimum standards. “Not relevant (+0.5)”
- No ranking (+0), One not presenting CORE ranking.

The JCR[2] published by Thomson Reuters[3] offers annual rankings of science and social science journals in the subject categories relevant to the journal, based on Impact Factor (IF) results. In each subcategory, quartile rankings are obtained

1∗ http://portal.core.edu.au/conf−ranks/

2∗∗ https://www.scimagojr.com/journalrank.php?category=1710

3∗∗http://mjl.clarivate.com/cgi-bin/jrnlst/jlsearch.cgi?PC=MASTER&Error=1

from each journal by occupying the quartile of the IF distribution in the same subject category.

Q1 - Denotes the top 25% of the IF distribution. "Very relevant (+2)"

- Q2 – Denotes middle and high positions (between top 50% and top 25%). "Relevant (+1.5)"
- Q3 –Denotes Middle-low position (top 75% to top 50%). "Somewhat Relevant (+1)"
- Q4 –Denotes the lowest position (bottom 25% of the IF distribution). "Not relevant (+0.5)"
- No ranking (+0), One not in the JCR list.

a) The primary interest of the paper is the security approach used in the SSD context. Yes (+1), Partially (+0.5), and No (+0).

b) The paper presents and/or explicitly assesses an approach as a solution to deal with software security risks. The study obtains the full score (Yes (+1)) if it presents a new approach, (Partially (+0.5)) if it presents an existing approach, and (No (+0)) if it does not present any solution to deal with software security risks.

c) The study is empirical and presents relevant data for our SMS. The results and conclusions of the study are strengthened by empirical evidence and it provides important and reliable information about future research and practice [76]. Yes (+1), Partially (+0.5), and No (+0).

The classification scheme and its finding are presented in Appendix A.

G. DATA EXTRACTION AND SYNTHESIS

We have extracted the following data from each article of final selection (116 papers):

- Paper-ids and Publication Title (presented in Appendix-A)
- Security Methods (RQ1.1, presented in Table 3)
- SDLC Phase discussed and covered (RQ1.2, presented in Section V, Subsection B, Figure 3)
- SWOT Analysis of Security Approaches (RQ1.4, presented in Section V, Subsection C)
- Publication Channel (RQ1.4, presented in Section V, Subsection D, Table 4, 5)
- Quality Assessment of Papers (presented in Appendix-A)
- Paper Titles (presented in Appendix-B)
- Paper Publication Year (presented in Appendix-A)
- Research Method used in Paper (presented in Appendix-A)
- Security Approach Types discussed in Paper (presented in Appendix-A)

V. RESULTS AND DISCUSSION

To answer the research questions presented in Section IV, we have shared the results of this study in these sections as:

Appendix A shows that 78.5 percent of studies ranked either higher than or equal to the average score of 2.5 points. Systematic reviews, therefore, emphasize that evaluating the quality of the study selection is important. The quality assessment is an important step in gaining a general view of the consequences of the paper on the subject [77].

A. WHAT ARE THE SECURITY METHODS IMPLEMENTED BY SOFTWARE DEVELOPMENT ORGANIZATIONS THAT RETURN THE LARGEST COMPETITIVE EDGE?

To answer RQ1.1, we extracted data by considering the presence of each SSE method. The results of the collected data according to this question are highlighted in Table 3.

In this SMS "Software security measurement and analysis" is the most cited (29%) category. CORAS [78] is the most popular method in this category. CORAS works as a model-driven method used towards the defensive risk analysis. Another most cited (6%) SSE method in this category is 'Appropriate-and-Effective-Guidance-for-Information-Security (AEGIS)'. AEGIS [79] also works as a software engineering method to create secure systems based on security requirements identification upon context used, modeling, and risk analysis.

The term "Privacy-preserving Technology" has the next highest frequency value (26%). In this category "SecureUML" and "UMLsec" is the most cited (17%) SSE methods, where SecureUML is an approach aiming to bridge the gap amongst the security-modeling-languages and design-modeling-languages, whereas UMLsec works as a UML extension allowing the characteristics of security-relevant to access control and confidentiality [84].

The next highest frequency value, in the study, is "Software testing" methods (24%). In this category "Software static analysis" with a frequency 7 is the most popular software testing methodology. Another dominant category in this study is "Secure requirement engineering" (16%). Table 3 shows that 10 studies in this SMS considered the "Security Quality Requirement Engineering (SQUARE) method". For eliciting and documenting the security requirements, the SQUARE method is used [80].

B. WHICH PARTICULAR SDLC PHASE HAS BEEN MOST DISCUSSED AND ADDRESSED IN THE LITERATURE?

This research question focus to find those studies in which a particular software development life cycle phase covered the security aspect. According to our findings, the phase of SDLC where the security aspect is considered may vary from study to study. Figure 3 shows that 32% of studies in our SMS highlighted software security methods in the design phase of the SDLC. This is typically from the fact that the vulnerabilities of design level works as the major sources of security risks in software systems [86], [87]. 50% of software defects are usually identified and detected in the designing phase of SDLC [81]. For developing secure software systems, the design phase of SDLC works as the basis in this regard [87]. Reducing the risks at this phase may minimize the efforts in other phases. It is mandatory for a software designer to properly check the aspect of security

TABLE 3. Secure software engineering methods.

Categories	Freq	% (N=116)	Secure Software Engineering Methods	Paper ids (mentioned in papers)	Freq
Software Testing	28	24	Software static analysis	10, 11, 89, 90, 98, 108, 116	7
			Penetration testing	9, 18, 40, 86, 94	5
			Detailed scenario-based test cases	4, 22, 24	3
			Dynamic security testing	98, 108	2
			Functional and Performance testing	4	1
			Adhoc testing	4	1
			Gray box testing	9	1
			Syntax testing	22	1
			Fault injection	22	1
			Mutation testing	22	1
			Gligor's testing method	22	1
			Unit Testing for Security	29	1
			Integration testing	32	1
			PACTs: Process controls and Tests	68	1
			TSP (Thrift Saving Plan) Secure	30	1
Secure Requirement Engineering	18	16	SQUARE: Security Quality Requirements Engineering method	17, 30, 36, 37, 38, 46, 47, 51, 64, 85	10
			SREP: Security requirement engineering process	38, 46, 51, 110	4
			SecREAD: Security Requirement Elicitation, Assessment and Design methodology	2, 40, 102	3
			SecReq: Security requirement engineering methodology	84	1
Software Security Measurement and Analysis	34	29	CORAS works as a model-driven-method for the defensive-risk-analysis	1, 20, 64, 75, 91, 110, 115	7
			AEGIS: Appropriate-and-Effective-Guidance-for-Information-Security	17, 20, 30, 51, 72, 76,	6
			OCTAVE (Operationally-Critical-Threat-Asset-and-Vulnerability-Evaluation)	21, 47, 82, 92, 115	5
			CRAMM (CCTA-Risk-Analysis-and-Management-Method)	16, 46, 115	3
			CRAMM (CCTA-Risk-Analysis-and-Management-Method)	16, 46, 115	3
			Magerit: Information-system-risks-analysis-and-management-methodology	82,	1
			SafSec Methodology: a unified approach to risk assessment for the SAFety-and-SECurity	19	1
			ISSRM: Information System security risk management	75,	1
			Common Weakness Enumeration typology to generate speculative abuse cases.	9	1
			Goal-Tree-Success-Tree and Master-Logic-Diagram (GTST-MLD), proposed to model the software-development-life-cycle to ensure the software quality on the basis to meet the criteria of high integrity and safety systems.	31	1
			FADES: formal analysis and design for engineering security, goal-oriented s/w security eng approach	52	1
			ACSM/SAR (Adaptive-Countermeasure-Selection-Mechanism/Security-Adequacy-Review)	16	1
			ACSM/SAR (Adaptive-Countermeasure-Selection-Mechanism/Security-Adequacy-Review)	16	1
			software security measurement and analysis (SSMA)	12	1
			STPA: Systems-Theoretic-Process-Analysis is one hazard analysis technique	68	1
Privacy-preserving Technology	30	26	SecureUML, and UMLsec is a methodology to model the access control policies and integrating them into the model-driven development.	7, 16, 18, 20, 30, 36, 38, 39, 40, 82, 84, 85, 86, 94, 95, 96, 102, 106, 107, 108, 110, 114, 115	19
			RAPPOR (is an attractive privacy-preserving-technology to monitor client database	6	1
			BDMP: Boolean-logic-Driven-Markov-Process, combine the safety and/or security aspects	64	1
			The data can be combined and collected through elaborated cryptographic method .i.e. partially-homomorphic-Paillier encryption or the users upon providing trace data can rely simply on Tor-like-network anonymization	6	1

TABLE 3. ***(Continued.)*** **Secure software engineering methods.**

			Software watermarking and code obfuscation is amongst the important methods for software security.	17	1
			AsmLSec [17], and UMLintr [23])	30	1
			XACML: eXtensible Access Control Markup Language	95	1
			SGM: Security goal model language	70	1
			Secure xADL	96	1
			SecureSOAmodelling language	96	1
			The multilevel object modeling technique	42	1
			SysML is prominent object-oriented modeling languages	107	1
Agile Methodology	2	2	Feature Driven Development (FDD) method	25	1
			FDSD: Failure-driven software design	80	1
Others	16	13	Petri-nets can be utilized to model attacks	13, 94, 95, 106, 111	5
			Common Attack Pattern Enumeration and Classification (CAPEC)	9, 11, 30, 37	4
			BIBIFI: Build-it, Break-it, Fix-it, a new security contest	77	1
			PSSS (Process to Support Software Security)	21	1
			ABD: Assurance Base development	104	1
			Stochastic maintenance method	34	1
			Wireframing	4	1
			The faceted scheme description provides a partial solution for the classification and/or retrieval problem. Policy Facet, Requirements Facet. Risks Facet, Trust Facets, Specification Facet, Security Risk Management Facet, Modeling Facet, Verification Facet, Purpose Facet, Coverage Facet, Alignment Facet, Granularity Facet, Flexibility Facet, Iteration Facet, Enactment Support Facet, Development Life-cycle Coverage Facet, Change Support Facet	8	1
			Teoriya-Resheniya-Izobretatelskikh-Zadatch (TRIZ) also known as Theory-of-Inventive-Problem- Solving (TIPS), is an approach and model-based-technology to generate innovative ideas and/or solutions to problem-solving.	34	1

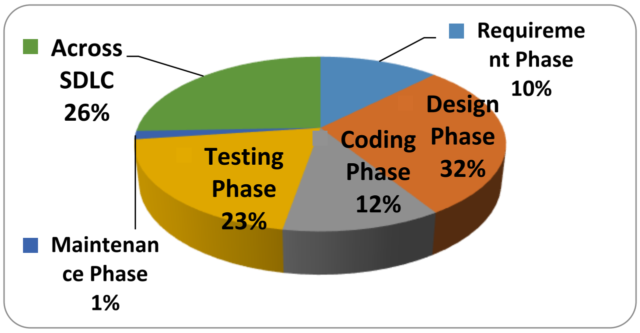


FIGURE 3. **Frequency of software security studies in SDLC.**

during this phase and immediately make the necessary changes that will ultimately improve the overall security. Software security attributes play a key role in security design [37]. These attributes help to enhance software security and also help to ensure software quality [82].

In this study, we have noted that 26% of studies considered software security across the whole SDLC. This fact is rising due to the integration of security in all phases of SDLC, which is a challenging issue in most organizations. We argued that the consideration of security should be an effective input towards the other phases of SDLC i.e. from requirements gathering to designing, implementation, testing, and deployment. For designing, building, and deploying secure software, it is essential to adapt the current software engineering practices and methodologies for the inclusion of specified security-related activities.

Figure 3 portrays that 22% of the selected studies considered the security of software in the testing phase of SDLC. The most lengthy, complex, and expensive phase of SDLC is software testing [83]. Software testing is a vital activity for increasing the quality of software development projects. It is a core phase for software development, but to test the programs thoroughly is not always the core subject under software engineering education [91]. Due to this shortfall, the software developers frequently treat the software testing as a liability which finally affects the overall quality of software. One underlying reason is that standardized testing mechanisms are frequently perceived as boring and challenging as compared to the creative coding phase and design activities [91].

The security testing technique is one of the most significant, effective, and commonly applied measures for the improvement of software security, used to identify the vulnerabilities and to ensure the functionality of security. For the identification of threats that might compromise security, threat modeling is a systematic way in this regard and it has been considered a well-known accepted practice by the software testing industry [92].

Figure 3 shows that 12% of studies in this SMS highlighted security approaches in the coding phase of the SDLC. Deployed software is continuously under the attack of hackers exploiting vulnerabilities for decades and an increase is being noticed in these attacks [39], [84]. The antiviruses, intrusion detection mechanisms, and firewalls are not enough to solve this problem. Each phase of SDLC must include various suitable security defenses, analysis, and countermeasures that result in further secure released code [85], [86]:

- Encrypt the files and copies. Storing data in encrypted form helps all output and backup versions of databases to be secured. Data Sunrise Data Encryption is the only way to do so.
- Auditing all the servers and copies. Using so helps you see who's been trying to get access to confidential info.
- Using the Intrusion Detection System (IDS) network.
- It is recommended that a stringent access and privilege management policy be enforced and maintained.
- Don't give client staff excessive rights and revoke expired rights in time.
- Secure coding is the practice of developing computer software in a way that guards against the accidental introduction of security vulnerabilities.
- Secure coding standards are rules and guidelines used to prevent security vulnerabilities. Used effectively, these security standards prevent, detect, and eliminate errors that could compromise software security.
- Validate input. Validate input from all untrusted data sources
- Heed compiler warnings
- Architect and design for security policies
- Keep it simple
- Default deny
- Adhere to the principle of least privilege
- Sanitize data sent to other systems
- Practice defense in depth
- Use effective quality assurance techniques

It is therefore needed from requirement-to-design-to-implementation-to-testing and deployment, software security necessarily is integrated throughout SDLC to provide the secure and best software to the user community.

We have noted that 12 out of 116 (10%) studies in the SMS have discussed security methods in the requirement phase of SDLC. Nowadays, where most software in the cloud and web-based, facing a diversity of stakeholders with entangled requirements, the development of secure software is a complex task.

During software development, security is often neglected. Nowadays, the researcher's emphasis to include the security aspects in every phase of software development during SDLC, specifically at the early phases. Goel *et al.* [95], suggests Security-Requirements-Elicitation-and-Assessment-Mechanism (SecREAM) holds the security issues that appear at the start of software development. Mead [87] provides a mechanism for measuring the security requirements engineering process, aligned with the method of Security-Quality- Requirements-Engineering (SQUARE). Similarly, in the maintenance phase of SDLC, we aimed to find any security-related mechanism in the software. Song-Kyoo [88] suggested the stochastic type maintenance method for the security of software through the use of a closed Queuing-model of unreliable backups.

C. WHAT IS THE SWOT ANALYSIS OF SECURITY APPROACHES IN SECURE SOFTWARE DEVELOPMENT?

Each of the above approaches contributes to its benefits and limitations in a specific domain. In the following subsections, we presented the SWOT (Strength, Weakness, Opportunity, Threat) analysis of security approaches in details:

1) MODELS

a: SSD-CMM: Systems Security Engineering Capability Maturity Model

Strength: SSD-CMM covers all phases of the software development process.

Weakness: It does not guarantee good results:

- Appraisal similarities.
- You need to properly understand the model and how to use it.
- It does not remove the need for testing/evaluation.
- Understanding how SSE-CMM contributes to the validation.

Opportunity: SSE-CMM looks for the following security features:

- Operational security
- Information security
- Network security
- Physical security
- Staff security
- Administrative security
- Communication security
- Security of meetings.

Threats:

- Requires great Investment
- Requires commitment at all levels
- The need to translate PAs into the context of the organization

b: Building Security In Maturity Model (BSIMM) [89]

Strength: The BSIMM project provides SSDL maturity and control risk

Weakness: Not secure for all uses.

Opportunity: It provides an opportunity concerned with measuring a large number of vendors to manage security risks and threats

Threats: IT does nothing to guarantee that any particular vendor product is secure

c: Misuse case modeling [90]

Strength: It focuses on security from the beginning of the design process and provides early design decisions.

Weakness:

- Its main weakness is that it is simple.
- It needs to be combined with powerful tools to establish an adequate project implementation plan.

Opportunity: It allows us to provide equal weight for operational and non-operational needs (e.g. requirements, platform requirements, etc.), which may not be possible with other tools.

Threats: Lack of structure and semantics.

d: Secure Tropos Methodology [91]

Strength: This approach incorporates the requirement engineering concepts of need such as character, purpose, planning, and security engineering concepts such as threat, security issues, and security approach under a cohesive security protection process.

Weakness: Modeling IS and Business assets are confusing.

Opportunity: It is a way to develop security-conscious software to support the analysis and development of secure and reliable software programs.

Threats: Do not include modeling constructs to model the attacker's intention and the threat impacts.

e: McGraw's Secure Systems Development Life Cycle (SSDLC) model [22]

Strength: SSDLC is a step-by-step method of developing information systems. Typical tasks include:

- Budget cutting
- Collecting business needs
- Modeling and
- Writing user documentation

Weakness: Documentations are expensive and time-consuming to make. It's also hard to keep right now.

Opportunity: SSDLC provides security in developing information systems.

Threats: Secure software development (SSD) activities for implementation: None

f: RBAC: Role-Based Access Control Model [92]

Strength: Restricting the access of the system to registered users. Mandatory access control (MAC) or Discretionary Access Control (DAC) can be enforced.

Weakness:

- It would be difficult if you try to build a complicated role for a large organization as there will be thousands of workers with few positions that can trigger a role explosion.
- Some restrictions can be made on accessing certain device activities using RBAC, but you cannot limit access to certain data.
- You can delegate permissions and privileges to user functions, but not to operations and objects.

Opportunity:

- Reduce administrative work and IT support.
- With RBAC, when an employee is hired or changes their job, you can decrease the need for paperwork and password changes.
- Maximizing operational efficiency.
- RBAC offers a streamlined approach that is logical in the definition.
- Improving compliance.

Threats:

- Role Explosion
- Security Risk Tolerance
- Scalability and Dynamism
- Expensive and
- Difficult Implementation

2) FRAMEWORKS

a: Trustworthy Computing Security Development Life Cycle or Microsoft Software Development Life Cycle (MS SDL) [93]

Strength: MS SDL is policy compliant system that:

- Assesses security policy compliance
- Meets security policy
- Provides code security training
- Establishes security requirements
- Conducts a final security review, and
- Executes an incident response plan

Weakness: The process is not perfect and is still evolving.

Opportunity:

- MS SDL performing dynamic analysis
- Establish design requirements
- Using threat modeling, and
- Approved tools

Threats: The process may either to reach perfection or to cease evolving in the foreseeable future.

b: KAOS: Knowledge acquisition in automated specification Framework [94]

Strength: It offers a comprehensive and sound way of arranging security requirements and using a graphical way of resolving security risks.

Weakness:

- One of the problems associated with KAOS with the transformations between models of different approaches is information loss.
- There is no graphical concrete syntax in the statement part of the acquisition language that would represent the different concepts and links supported by the meta-model.
- Deontic logic extensions are not well incorporated into the statement sublanguage to support a deeper level of systematic reasoning regarding agent capabilities and assignment of responsibilities.
- More explicit support should also be given to cooperation and communication among agents.

Opportunity: In different fields such as mechanics, telecommunications, healthcare, KAOS finds its application and can therefore be used for any kind of information system.

Threats: Lack of inbuilt mechanism for evaluating security risks.

c: SECTET: Model-Driven Security Engineering framework [95]

Strength: This approach allows the automated generation of security policies and workflow used on the target framework for the configuration of security services.

Weakness: This method does not currently use as a user interface to write high-level security policies.

Opportunity: Moreover, the main benefits of this approach are the separation of security-enhanced business modeling, security modeling, code creation, and system execution.

Threats: Lack of high-level security policies

3) METHODS

a: SQUARE: Security Quality Requirements Engineering method [96]

Strength: It integrates security considerations into the early stages of SDLC.

Weakness: SQUARE did not achieve the following activities:

- The identification of users of the software.
- Detection of possible device attackers.
- Recognition of the attacker's interest in a piece of software's resources/assets.
- Detection of the capabilities of the attacker.

Opportunity: SQUARE offers a way for information technology systems and applications to elicits, categorize, and prioritize security requirements.

Threats: Lack of identification of an illegal access

b: SecureUML, and UMLsec [97]

Strength: UMLsec makes it possible to model security features related to confidentiality and access control.

Weakness: UMLsec is guided by security criteria, however, it does not mean to model them explicitly.

Opportunity: The goal of SecureUML is to bridge the gap between languages for security modeling and languages for design modeling.

Threats: Only permitted activities were executed.

There is no required unique restriction language in UMLsec.

c: CORAS: Combine methods for risk analysis and semiformal description methods [98]

Strength: For modeling risks and unwanted actions, the CORAS profile provides advanced case diagrams for usages.

Weakness: CORAS just draw threat scenarios/diagram.

Opportunity: It encourages engagement and cooperation between various groups of stakeholders involved in a risk assessment.

Threats: CORA's threat modeling tool has no risks/threats analysis facilities.

d: AEGIS: Appropriate and effective guidance for information security [99]

Strength: Identification of properties, abuse cases, operational context, and support for simulation of security requirements.

Weakness: Secure software development (SSD) activities for implementation: None

Opportunity: For the creation of stable and functional systems, AEGIS offers essential resources

Threats: There are no SSD security assurance activities and Usage in the Industry is also not reported

e: OCTAVE (The Operationally Critical Threat, Asset, and Vulnerability Evaluation) [100]

Strength: Decisions are focused on the risks of confidentiality, the credibility of sensitive data-related assets

Weakness: It may still be useful outside of their boundaries, the result cannot be guaranteed.

Opportunity: Organizational risks are calculated by OCTAVE and focus on strategic and functional aspects.

Threats: Lack of trust.

f: Petri nets [36]

Strength: For modeling attacks, we can use the Petri net.

Weakness: Petri net model has weaknesses in its inability to test an unbounded position for exactly a particular marking and to take action on the result of the test.

Opportunity: It is one of several mathematical modeling languages for the description of a distributed system. The continuity of events has become increasingly common.

Threats: Due to its concurrency, it increases the complexity of operations.

4) PROCESSES

a: Aprville and Pourzandi's Secure Software Development Life Cycle Process [55]

Strength: This process provides:

- Identification of high-level security
- Modeling of threats
- Specification of security requirements, and
- Risks analysis

Weakness: Usage in the Industry: Not reported

Opportunity:

- Avoiding vulnerabilities of buffer overflow and a format string
- Usage of established security algorithms
- Usage of secure programming language

Threats: This process mostly does not use the artifacts of SSD activity In the later stage of software development.

b: Software Security Assessment Instrument(SSAI) [101]

Strength: SSAI detects vulnerabilities and also provide mitigation practices for it.

Weakness: No SSD activities for design, implementation. Usage in the industry is none.

Opportunity:

- SSAI mitigate vulnerabilities
- It provides a list of scanning tools for the security code.
- It offers potential security checklist items to guide software development.

Threats: This process also mostly does not use the artifacts of SSD activity In the later stage of software development.

TABLE 4. Articles by their journal and conference rank.

Journals	Frequency	% (N=116)	Conferences, Workshops, Symposiums	Frequency	% (N=116)
Q1	8	6.9 %	CORE A*	10	8.6 %
Q2	30	25.9 %	CORE A	3	2.6 %
Q3	4	3.4 %	CORE B	7	6 %
Q4	4	3.4 %	CORE C	8	6.9 %
No Ranking	11	9.5 %	No Ranking	31	26.7 %

TABLE 5. Top publication venues of identified articles.

Publication Channels	Freq	% (N=116)	Publication Venues	Freq
Journals	57	43.9 %	Information and Software Technology	6
			Software System Model	5
			Journal of System and Software	4
			Computer Standards & Interfaces	4
			Security and Communication Networks	3
Conferences	42	36.2 %	International Conference on Availability, Reliability, and Security	5
			International Conference on Software Engineering	4
Workshops	9	7.7 %	Type: Software Security and Privacy	8
			Type: Software Engineering	1
Symposiums	8	6.9 %	Type: Software Security and Privacy	4
			Type: Software Engineering	2

D. WHAT ARE THE POPULAR VENUES FOR SECURE SOFTWARE DEVELOPMENT?

Table 4 presents information about the total score of the selected articles based upon their considerations from Computing Research Education (CORE)*2018 and Journal Citation Reports (JCR)**2018 lists.

The table data shows that 64% of the selected papers in our study are published in Q1-Q4 journals and Core A*-C conferences. The remaining 36% of articles are published in well-reputed conferences and journals; however, some of them have no ranking in the CORE 2018 and JCR lists. The detailed score for each of the selected studies is available in Appendix B.

Table 5 shows the distribution of selected studies based on publication types. Journals, conferences, workshops, and symposiums are the four main publication types with 43.9% (57 studies), 36.2% (42 studies), 7.7% (9 studies), and 6.9% (8 studies) of the selected studies respectively. Overall, 116 publications' venues are identified that cover various areas of computer science; such as software engineering, software security, and networking, etc. It means that such a domain has received large attention from the researchers. We observed that "Information and Software Technology", "Software System Model", "Journal of System and Software", "Computer Standards and Interfaces", and "Security and Communication Networks" are the most recurring publication venues for security approaches for SSD topic. We also explored that "International Conference on Availability, Reliability, and Security" and "International Conference on Software Engineering (ICSE)" is the most frequent source of worthy publications to the topic concerned. It demonstrates the importance of software security research in software engineering and other related fields.

VI. IMPLICATIONS FOR RESEARCH AND PRACTICE

The findings of this research contribute to the SSD context from various perspectives. Initially, they provide a clearer understanding of the security practices and methods used in the SSD environment for the academic community and explain differences in areas where security approaches are ineffective or inadequate. This knowledge will open opportunities for SSD researchers to solve problems and design new security techniques that can minimize the impact of security risks and threats in the development of secure software. Secondly, the development of applications participating in highly distributed projects would provide an overview of current approaches aimed at promoting their efforts and recognizing those most likely to meet their needs. The guidelines taken from the conclusions of this SMS for researchers and practitioners are as follows:

- Practitioners, such as software developers and project managers, are eager to learn about the latest research on the subject, along with researchers interested in SSD project management. They need to study the published articles more from the "Information and Software Technology", "Software System Model", "Journal of System and Software", "Computer Standards and Interfaces", and "Security and Communication Networks" journals. Similarly, it will be more beneficial for them to read the published articles from the proceedings of the "International Conference on Availability, Reliability, and Security" and "International Conference on Software Engineering (ICSE)" and its affiliated workshops and symposiums. In short, we can say that the above venues are important sources to study the security approaches in SSD. Similarly, researchers are encouraged to submit their high-quality research

TABLE 6. Classification of selected studies based on data extraction.

P-id	Publication Channel	Publication Year	Research Method	Security Approach	Quality Assessment				
					A	B	C	D	Score
1	Conference	2011	Formal Methods	Model and Tool	0.5	1	0.5	0	2
2	Conference	2015	Case Study	Method	0	1	1	1	3
3	Conference	2009	Experiment	Processes and Tools	0	1	1	0.5	2.5
4	Conference	2015	Ordinary Literature Review	Model	0	1	0.5	0.5	2
5	Journal	2014	Experiment	Tool	1.5	1	1	1	4.5
6	Symposium	2016	Experiment	Models and Methods	1.5	1	1	1	4.5
7	Conference	2011	Experiment	Tool	2	1	1	0.5	3.5
8	Conference	2007	Ordinary Literature Review	Framework	0	1	0.5	0.5	2
9	Symposium	2016	Case Study	Method and Tool	0.5	1	1	1	3.5
10	Conference	2010	Experiment	Method	0	1	1	0.5	2.5
11	Conference	2013	Experiment and Case Study	Metrics	2	1	1	0.5	4.5
12	Conference	2012	Case Study	Method	1	1	1	1	4
13	Conference	2006	Experiment and Case Study	Model	1.5	1	1	1	4.5
14	Conference	2009	Case Study	Process	0.5	1	1	1	3.5
15	Journal	2005	Ordinary Literature Review	Portal and Tools	1.5	1	0	0	2.5
16	Journal	2004	Experiment	Framework	1.5	1	1	0	3.5
17	Conference	2012	Experiment	Method	0.5	1	1	1	3.5
18	Conference	2009	Experiment	Process	1.5	1	1	1	4.5
19	Conference	2011	Ordinary Literature Review	Framework	0	1	0.5	0.5	2
20	Conference	2008	Ordinary Literature Review	Methods and Tools	1	1	0	1	3
21	Conference	2009	Experiment	Process	1	1	1	1	4
22	Conference	2002	Ordinary Literature Review	Guidelines	1.5	0.5	0	0	2
23	Workshop	2007	Ordinary Literature Review	Process and Model	0	1	0	0.5	1.5
24	Conference	2015	Systematic Literature Review	Principles and Guidelines	1	0.5	0	0.5	2
25	Journal	2013	Case Study	Process	2	1	1	1	5
26	Conference	2015	Ordinary Literature Review	Guidelines	0	0.5	0	0	0.5
27	Conference	2011	Case Study	Process and Method	0	1	0.5	1	2.5
28	Conference	2016	Ordinary Literature Review	Model	0	1	0.5	0.5	2
29	Conference	2007	Experiment	Process	0	1	0.5	1	2.5
30	Conference	2009	Ordinary Literature Review	Process and Method	1	1	1	1	4
31	Symposium	2005	Experiment	Method	0	1	1	0.5	2.5
32	Conference	2013	Experiment	Tool	0	1	1	1	3
33	Conference	2008	Case Study	Patterns and Metrics	1	0.5	0.5	1	3
34	Conference	2009	Case Study	Method and Tool	0	1	1	1	3
35	Journal	2012	Experiment	Method	1.5	1	1	1	4.5
36	Journal	2011	Ordinary Literature Review	Framework	1.5	1	0.5	1	4
37	Conference	2016	Systematic Mapping Study	Processes, Methods, Models and Frameworks	0	1	0	1	2
38	Journal	2012	Survey	Methods	1.5	1	0.5	1	4
39	Conference	2015	Experiment	Framework	2	1	1	0	4
40	Journal	2016	Ordinary Literature Review	Process	0	1	0	1	2
41	Journal	2007	Ordinary Literature Review	Models	1	1	0	1	3
42	Journal	2005	Survey	Methods	2	1	0.5	1	4.5
43	Journal	2015	Experiment and Case Study	Model and Tool	2	1	1	1	5
44	Journal	2009	Ordinary Literature Review	Models	1.5	1	0	1	3.5
45	Journal	2006	Formal Methods and Case Study	Framework and Tool	1.5	1	1	1	4.5

TABLE 6. ***(Continued.)* Classification of selected studies based on data extraction.**

46	Journal	2007	Ordinary Literature Review	Process	1.5	1	0	1	3.5
47	Journal	2010	Systematic Literature Review	Techniques, Tools, Frameworks, Processes, Models, Methods	1.5	1	0	1	3.5
48	Journal	2005	Case Study	Method	2	1	1	1	5
49	Journal	2015	Systematic Literature Review	Techniques, Tools, Frameworks, Processes, Models, Methods	1.5	1	0	1	3.5
50	Conference	2008	Ordinary Literature Review	Guidelines and Recommendations	0	0.5	0	1	1.5
51	Journal	2016	Systematic Mapping Study	Tools, Frameworks, Processes, Models, Methods	1.5	1	0	1	3.5
52	Journal	2009	Formal Methods and Surveys	Model and Tool	1	1	1	1	4
53	Journal	2017	Experiment	Strategies, Tools and Metrics	1.5	0.5	1	1	3.5
54	Conference	2009	Experiment and Survey	Tool	1	1	1	1	4
55	Conference	2014	Survey	Processes	0	1	0.5	1	2.5
56	Journal	2015	Systematic Literature Review	Notations	1.5	0.5	0	1	3
57	Journal	2015	Ordinary Literature Review	Process	1	0.5	0.5	1	3
58	Conference	2011	Ordinary Literature Review	Guidelines and Recommendations	0.5	0.5	0	1	2
59	Journal	2009	Surveys	Process and Tool	2	1	0.5	1	4.5
60	Journal	2014	Experiment	Model and Tool	1.5	1	1	1	4.5
61	Journal	2009	Experiment	Tool	1.5	1	1	1	4.5
62	Journal	2008	Surveys	Guidelines	0	0.5	0.5	1	2
63	Journal	2007	Case Study	Process	1.5	1	0.5	1	3.5
64	Journal	2012	Ordinary Literature Review	Methods	2	1	0	1	4
65	Journal	2016	Case Study	Model	0	1	1	1	3
66	Symposium	2017	Experiment	Tools	2	1	1	0.5	4.5
67	Workshop	2006	Experiment	Process	0	1	1	0.5	2.5
68	Conference	2015	Systematic Mapping Study	Framework	2	1	1	1	5
69	Conference	2015	Ordinary Literature Review	Model	0.5	1	1	1	3.5
70	Workshop	2010	Experiment	Method	2	1	1	1	5
71	Workshop	2004	Experiment	Method and Framework	0.5	1	1	1	3.5
72	Workshop	2003	Experiment	Method	0.5	1	1	1	3.5
73	Workshop	2008	Ordinary Literature Review	Framework and Model	0	1	0	0	1
74	Symposium	2010	Experiment	Method	0	1	1	0.5	2.5
75	Journal	2015	Ordinary Literature Review	Guidelines and Recommendations	0	0.5	0	0.5	1
76	Symposium	2009	Ordinary Literature Review	Guidelines	2	0.5	0	0.5	3
77	Conference	2016	Experiment and Survey	Method	0	1	1	1	3
78	Conference	2005	Experiment	Models	0	1	1	1	3
79	Conference	2010	Ordinary Literature Review	Methods and Tools	2	1	0	1	4
80	Conference	2007	Ordinary Literature Review	Process and Tool	2	1	1	1	5
81	Conference	2017	Systematic Mapping Study	Guidelines, Processes, Methods	0	0.5	0	1	1.5
82	Journal	2006	Ordinary Literature Review	Models and Methods	0	1	0	1	2
83	Journal	2007	Ordinary Literature Review	Model	0	1	1	1	3
84	Journal	2014	Ordinary Literature Review	Methods	1.5	1	0	1	3.5
85	Journal	2010	Ordinary Literature Review and Case Study	Framework	1.5	1	1	1	4.5

TABLE 6. ***(Continued.)*** **Classification of selected studies based on data extraction.**

86	Symposium	2004	Survey	Model	1	1	1	0	3
87	Journal	2012	Experiment and Survey	Process	1.5	1	1	1	4.5
88	Journal	2012	Case Study	Tool	1.5	1	1	1	4.5
89	Journal	2012	Survey	Metrics	1.5	1	0.5	1	4
90	Journal	2008	Ordinary Literature Review	Models, Methods, Tools	0	1	0	1	2
91	Journal	2016	Systematic Literature Review	Models	2	1	0	1	4
92	Symposium	2004	Experiment	Algorithm and Tool	0	1	1	0.5	2.5
93	Journal	2016	Ordinary Literature Review and Case Study	Guidelines and Recommendations	1.5	0.5	0	1	3
94	Journal	2012	Case Study	Models	1.5	1	0.5	1	4
95	Journal	2012	Experiment and Survey	Models	1.5	1	0.5	1	4
96	Journal	2015	Formal Methods	Framework	1.5	1	1	1	4.5
97	Journal	2018	Systematic Mapping Study	Metrics	1.5	1	0	1	3.5
98	Conference	2016	Ordinary Literature Review	Method	0	1	1	1	3
99	Journal	2012	Ordinary Literature Review	Guidelines and Recommendations	0	0.5	0	0.5	1
100	Journal	2016	Ordinary Literature Review	Guidelines and Recommendations	0	0.5	0	0.5	1
101	Journal	2010	Formal Method	Framework and Tool	0.5	1	1	0.5	3
102	Journal	2016	Survey	Method	0.5	1	1	1	3.5
103	Journal	2010	Ordinary Literature Review	Framework	0	1	0.5	0	1.5
104	Conference	2012	Ordinary Literature Review	Models and Tools	0	1	0	0.5	1.5
105	Journal	2012	Case Study	Model	0.5	1	1	1	3.5
106	Journal	2015	Ordinary Literature Review	Methods	1.5	1	0	1	3.5
107	Conference	2011	Case Study	Framework	0	1	1	0.5	2.5
108	Journal	2017	Systematic Literature Review	Methods	0	1	0	1	2
109	Journal	2015	Systematic Literature Review	Tools, Methods and Processes	1	1	0	1	3
110	Journal	2015	Survey	Processes and Methods	2	1	0.5	1	4.5
111	Workshop	2009	Experiment and Survey	Method	2	1	1	0.5	4.5
112	Workshop	2008	Case Study	Tool	0.5	1	1	0.5	3
113	Workshop	2008	Case Study	Metrics	0	0.5	0.5	1	2
114	Journal	2010	Experiment	Framework	1.5	1	1	1	4.5
115	Journal	2013	Experiment	Framework	0	1	1	1	3
116	Journal	2012	Experiment	Model	0.5	1	1	1	3.5

papers to these journals and conferences in the SSD domain.

- The outcome of this SMS suggests a lack of empirical evidence of the different suggested solutions. In the study, only 30.12% of the security approaches have been empirically validated.

Researchers are encouraged to review their security methods using experiments, case studies, and ordinary literature review on the first choice of research methods to obtain stronger qualitative and quantitative outcomes and to build approaches that can fulfill the demands of SSD projects.

- As shown in Appendix A and B, many approaches are intended to enhance the security in SSD. Practitioners, particularly software developers and designers may use such approaches to care about the security in SSD. While working in this domain, we put a suggestion that practitioners might cooperate with researchers in applying the security approaches in their practice.
- The SSD domain is still immature and sufficient research work is required, as this area has not been reached a solution-oriented stage, particularly on empirically evaluated solutions.

VII. THREATS TO VALIDITY

The validity of the study is concerned with its findings being trustworthy. The limitations are listed as follows for this SMS:

- **Construct Validity**

The construct validity refers to the degree of the analysis to which the operational measures represent what the RQs

TABLE 7. Titles of the final selected papers.

Paper ids	Title of the final selected papers in our Systematic Mapping Study
1.	Threat modeling using Formal Methods: A New Approach to Develop Secure Web Applications.
2.	Security Requirements Elicitation and Assessment Mechanism (SecREAM)
3.	Security-aware Software Development Life Cycle (SaSDLC) – Processes and Tools
4.	Exhaustive study of SDLC Phases and their Best Practices to create CDP Model for Process Improvement
5.	Gauging the Impact of FISMA on Software Security
6.	Data-driven Software Security: Models and Methods
7.	Automated Security Hardening for Evolving UML Models
8.	Towards a Comprehensive View of Secure Software Engineering
9.	Improving Penetration Testing Methodologies for Security-Based Risk Assessment
10.	Research on XML Based Static Software Security Analysis
11.	Automated Software Architecture Security Risk Analysis using Formalized Signatures
12.	Measuring The Software Security Requirements Engineering Process
13.	Threat-Driven Modeling and Verification of Secure Software Using Aspect-Oriented Petri Nets
14.	The Impact of Test Case Reduction and Prioritization on Software Testing Effectiveness
15.	A Portal for Software Security
16.	Risk Analysis in Software Design
17.	A Method of Software Watermarking
18.	Activity and Artifact Views of a Secure Software Development Process
19.	The Merging Trend of Software Security and Safety
20.	Secure Software Design in Practice
21.	A Knowledge Management Approach to Support a Secure Software Development
22.	Maintaining Software with a Security Perspective
23.	On the Secure Software Development Process: CLASP and SDL Compared
24.	Literature Review of the Challenges of Developing Secure Software Using the Agile Approach
25.	Extending the Agile Development Approach to Develop Acceptably Secure Software
26.	Secure by Design Approach to Improve Security of Object Oriented Software
27.	Risk management assessment using SERIM method
28.	Integrating Risk assessment and Threat modeling within SDLC process
29.	Security in Coding Phase of SDLC
30.	On Selecting Appropriate Development Processes and Requirements Engineering Methods for Secure Software
31.	Security on Software Life Cycle using Intrusion Detection System
32.	Design of a Tool for Checking Integration Testing Coverage of Object-Oriented Software
33.	Using security patterns to combine security metrics
34.	Design of Enhanced Software Protection Architecture by Using Theory of Inventive Problem Solving
35.	Evaluation of the Pattern-based method for Secure Development (PbSD): A controlled experiment
36.	A framework to support alignment of secure software engineering with legal regulations
37.	Software Development Initiatives to Identify and Mitigate Security Threats: A Systematic Mapping
38.	Survey and analysis on Security Requirements Engineering
39.	A Security Practices Evaluation Framework
40.	Analytical network process for software security: a design perspective
41.	Software requirements and architecture modeling for evolving non-secure applications into secure applications
42.	Secure information systems development e a survey and comparison
43.	Automated analysis of security requirements through risk-based argumentation
44.	Model-Driven Development for secure information systems
45.	Foundations for Designing Secure Architectures
46.	A common criteria based security requirements engineering process for the development of secure information systems
47.	A systematic review of security requirements engineering
48.	When security meets software engineering: a case of modelling secure information systems
49.	An Extensive Systematic Review on the Model-Driven Development of Secure Systems
50.	Best Practices for Software Security: An Overview
51.	Exploring Software Security Approaches in Software Development Lifecycle: A Systematic Mapping Study
52.	Formal analysis and design for engineering security automated derivation of formal software security specifications from goal-oriented security requirements

TABLE 7. ***(Continued.)*** **Titles of the final selected papers.**

53.	An empirical study to improve software security through the application of code refactoring
54.	Static Code Analysis to Detect Software Security Vulnerabilities - Does Experience Matter?
55.	Benchmarking SDL and CLASP lifecycle
56.	Design notations for secure software: a systematic literature review
57.	A process for mastering security evolution in the development lifecycle
58.	Software Architectural Considerations For The Development of Secure Software Systems
59.	Should software testers use mutation analysis to augment a test set?
60.	Specifying model changes with UML change to support security verification of potential evolution
61.	Development Life-cycle of Critical Software Under FoCaL
62.	Secure software development: Why the development world awoke to the challenge
63.	On the design of more secure software-intensive systems by use of attack patterns
64.	Comparing risk identification techniques for safety and security requirements
65.	Time for Addressing Software Security Issues: Prediction Models and Impacting Factors
66.	FOSS Version Differentiation as a Benchmark for Static Analysis Security Testing Tools
67.	Towards a Structured Unified Process for Software Security
68.	A Security Practices Evaluation Framework
69.	Synthesis of Secure Software Development Controls
70.	Unified modeling of attacks, vulnerabilities and security activities
71.	MAC and UML for Secure Software Design
72.	Bringing Security Home: A process for developing secure and usable systems
73.	Formal Derivation of Security Design Specifications from Security Requirements
74.	Secure Code Generation for Web Applications
75.	Guiding the selection of security patterns based on security requirements and pattern classification
76.	Research on Software Design Level Security Vulnerabilities
77.	Build It, Break It, Fix It: Contesting Secure Development
78.	A Threat-Driven Approach to Modeling and Verifying Secure Software
79.	Research on Software Security Awareness: Problems and Prospects
80.	Failure-driven Software Safety
81.	Towards the Integration of Security Practices in the Software Implementation Process of ISO/IEC 29110: A Mapping
82.	The risks analysis like a practice of secure software development. A revision of models and methodologies
83.	SecSDM: A Model for Integrating Security into the Software Development Life Cycle
84.	From misuse cases to mal-activity diagrams: bridging the gap between functional security analysis and design
85.	A framework to support alignment of secure software engineering with legal regulations
86.	Software Security Building Security In
87.	Security modeling for service-oriented systems using security pattern refinement approach
88.	Improving Software Security with Static automatic Code analysis in an Industry setting
89.	Measuring Systems Security
90.	Software Security
91.	Model-based security testing: a taxonomy and systematic classification
92.	Security Engineering: Systems Engineering of Security through the Adaptation and Application of Risk Management
93.	The practice of secure software development in SDLC: an investigation through existing model and a case study
94.	Unified threat model for analyzing and evaluating software threats
95.	A threat model-based approach to security testing
96.	A security framework for developing service-oriented software architectures
97.	Mapping the Field of Software Security Metrics
98.	A Secure Software Design Methodology
99.	Improving Security in Software Development Life Cycle
100.	Top Fifty Software Risk Factors and the Best Thirty Risk Management Techniques in Software Development Lifecycle for Successful Software Projects
101.	A Secure Software Pattern Generation with Design Level Information
102.	Security in Requirement and Design Phases
103.	Constructing Authorization Systems Using Assurance Management Framework
104.	Software Security: The Dangerous Afterthought
105.	Comparison of SETAM with Security Use Case and Security Misuse Case: A Software Security Testing Study
106.	Specification, verification, and quantification of security in model-based systems

TABLE 7. *(Continued.)* **Titles of the final selected papers.**

107.	Model-Based Systems Security Quantification
108.	A Systematic Review of Agent-Based Test Case Generation for Regression Testing
109.	A systematic classification of security regression testing approaches
110.	Investigating Security Threats in Architectural Context: Experimental Evaluations of Misuse Case Maps
111.	Security Test Generation using Threat Trees
112.	Evaluating the Cost Reduction of Static Code Analysis for Software Security
113.	Security Metrics for Source Code Structures
114.	A UML-based static verification framework for security
115.	A framework for development of secure software
116.	Model-driven Secure Development Lifecycle

expect to respond. Two variables can be easily defined in this SMS as a challenge to the construct validity. The research string used in the study is one, while the digital libraries checked are the other. We conducted a systematic search using a wide variety of words in the sample to further expand the research scope. For the validity to include the maximum relevant literature, the keywords of the study were included after detailed discussion and recommendations by the two authors. The choice of digital libraries for the studies was another risk to construct validity. The identifying of five digital libraries as the key sources in such a domain mitigated this risk.

- **Internal Validity**

In internal validity, the classification and decision to allocate a specific security approach to a particular category and method of software development may be evaluated. The classification was suggested to minimize this effect and the categorization process was performed by the two authors, while the final results were checked by the third one. Moreover, to allow the conclusions drawn from the results to be replicated, the steps and activities in this scheme have been clearly defined. The internal validity risks have been minimized such that all interested readers are encouraged to freely see the data retrieved from the papers of the studies displayed.

- **Conclusion Validity**

The validity of the conclusion is the degree to which the assumptions made about the relationships are rational and concerned with the potential to reproduce these results. The possibility of conclusion validity was a factor in such SMS that could lead to the conclusion of incorrect findings of a relationship in the data observed. Each step of the data collection, extraction, and analysis was validated by a systematic process and periodic reviews carried out by the participating researchers to reduce this threat. The explanation behind this move was that for similar research, the same procedure was done in the literature.

- **External Validity**

External validity includes how much it is possible to generalize the outcomes of a study. To diminish this threat, the ratio of security models, frameworks, methods, processes, guidelines, metrics, and tools have been included in this work.

VIII. CONCLUSION AND FUTURE WORK

At each level of SDLC, incorporating software protection is now a primary need for secure software development. SSE advocates that the security of software is an essential factor that should be evaluated during the early stages of SDLC [5]. We need to incorporate security features into our application development life cycle and adapt the latest SSE approaches to build and deploy a secure software system [6], [7].

We conclude from the above discussion that it is not good enough to secure the software system in post-development phases and there is a dire need to figure out better ways and means to secure the software system. It is our conviction that this paper will contribute by offering a taxonomy for the SSE-related research subject and will provide a breakdown of papers in the same field on each topic. The taxonomical and demographical information on a particular research topic, presented in this paper, will invite more opportunities to promote further research in this field.

We investigate the outcome of this mapping study to provide a reference direction for interested and vigilant researchers to address and explore new research trends and gaps and the subject domain. This study will provide a deep insight into the subject domain that could enrich and more update the software practitioners about security approaches, security limitations, and unresolved specific and general issues. After the final selection and analysis of 116 articles were selected and these were classified based on security approaches according to the quality assessment, software security method, SDLC phase has been most discussed and addressed, publication venue, and SWOT analysis. The results indicate that this domain is still immature and sufficient research work needs to be carried out particularly on empirically evaluated solutions.

Also, we plan to develop a Software Security Assurance Model (SSAM) [18] to assist vendor organizations to assess their readiness for secure software development. We will develop the SSAM model by using the output of future RQs, supervisor inputs, and guidance from existing studies [102]–[111]. The model will produce numerous evaluation reports, such as a list of security measurements and their solutions to be used in each phase of the SDLC by GSD vendor organizations. Our main aim is to answer the following research questions (RQs) in the future to achieve the above-mentioned goals:

RQ1: What are the security threats to the development of secure software products, as described in the literature and industrial survey, to be avoided by GSD vendor organizations?

RQ2: What are the practices to be implemented by GSD vendor organizations, as defined in the literature and industrial survey, to build secure software products?

RQ3: Is the proposed SSAM model practically robust in helping GSD vendor organizations in assessing their readiness to build secure software?

APPENDIX A

See Table 6.

APPENDIX B

See Table 7.

ACKNOWLEDGMENT

The findings achieved herein are solely the responsibility of the authors. In addition to this, all members of the Software Engineering Research Group, University of Malakand (SERG_UOM) are obliged to do so in general and both Dr. Nasir Rashid and Ghulam Murtaza in particular, for their reviews and suggested improvements.

REFERENCES

[1] 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.

[2] 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.

[3] M. Z. Gunduz and R. Das, "Cyber-security on smart grid: Threats and potential solutions," *Comput. Netw.*, vol. 169, Mar. 2020, Art. no. 107094.

[4] E. K. Szczepaniuk, H. Szczepaniuk, T. Rokicki, and B. Klepacki, "Information security assessment in public administration," *Comput. Secur.*, vol. 90, Mar. 2020, Art. no. 101709.

[5] L. Bracciale, P. Loreti, A. Detti, R. Paolillo, and N. B. Melazzi, "Lightweight named object: An ICN-based abstraction for IoT device programming and management," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5029–5039, Jun. 2019.

[6] 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 Trans. Inf. Forensics Security*, vol. 14, no. 9, pp. 2315–2330, Sep. 2019.

[7] G. McGraw, "Six tech trends impacting software security," *Computer*, vol. 50, no. 5, pp. 100–102, May 2017.

[8] J. Li, Y. Zhang, X. Chen, and Y. Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," *Comput. Secur.*, vol. 72, pp. 1–12, Jan. 2018.

[9] A. Sharma and M. P. Kumar, "Aspects of enhancing security in software development life cycle," *Adv. Comput. Sci. Technol.*, vol. 10, no. 2, pp. 203–210, 2017.

[10] M. Essafi, L. Labed, and H. B. Ghezala, "Towards a comprehensive view of secure software engineering," in *Proc. Int. Conf. Emerg. Secur. Inf., Syst., Technol.*, 2007, pp. 181–186.

[11] R. Syed, M. Rahafrooz, and J. M. Keisler, "What it takes to get retweeted: An analysis of software vulnerability messages," *Comput. Hum. Behav.*, vol. 80, pp. 207–215, Mar. 2018.

[12] A. K. Srivastava and S. Kumar, "An effective computational technique for taxonomic position of security vulnerability in software development," *J. Comput. Sci.*, vol. 25, pp. 388–396, Mar. 2018.

[13] D. Mellado, C. Blanco, L. E. Sánchez, and E. Fernández-Medina, "A systematic review of security requirements engineering," *Comput. Standards Interfaces*, vol. 32, no. 4, pp. 153–165, Jun. 2010.

[14] I. Velásquez, A. Caro, and A. Rodríguez, "Authentication schemes and methods: A systematic literature review," *Inf. Softw. Technol.*, vol. 94, pp. 30–37, Feb. 2018.

[15] Y. Lee and G. Lee, "HW-CDI: Hard-wired control data integrity," *IEEE Access*, vol. 7, pp. 10811–10822, 2019.

[16] Z. A. Maher, H. Shaikh, M. S. Khan, A. Arbaaeen, and A. Shah, "Factors affecting secure software development practices among developers—An investigation," in *Proc. ICETAS*, 2018, pp. 1–6.

[17] G. E. Rodríguez, J. G. Torres, P. Flores, and D. E. Benavides, "Cross-site scripting (XSS) attacks and mitigation: A survey," *Comput. Netw.*, vol. 166, Jan. 2020, Art. no. 106960.

[18] R. A. Khan and S. U. Khan, "A preliminary structure of software security assurance model," in *Proc. 13th Int. Conf. Global Softw. Eng.*, Gothenburg, Sweden, 2018, pp. 137–140.

[19] K. S. U. Khan, R. Ahmad, and I. M. Yazid, "Systematic mapping study protocol for secure software engineering," in *Proc. Asia Int. Multidisciplinary Conf. (AIMC)*, 2019, pp. 367–374.

[20] G. McGraw, "From the ground up: The DIMACS software security workshop," *IEEE Secur. Privacy*, vol. 1, no. 2, pp. 59–66, Mar. 2003.

[21] G. Hatzivasilis, I. Papaefstathiou, and C. Manifavas, "Software security, privacy, and dependability: Metrics and measurement," *IEEE Softw.*, vol. 33, no. 4, pp. 46–54, Jul. 2016.

[22] G. McGraw, "Software security," *IEEE Secur. Privacy*, vol. 2, no. 2, pp. 80–83, Aug. 2004.

[23] R. M. Parizi, K. Qian, H. Shahriar, F. Wu, and L. Tao, "Benchmark requirements for assessing software security vulnerability testing tools," in *Proc. COMPSAC*, 2018, pp. 825–826.

[24] N. M. Mohammed, M. Niazi, M. Alshayeb, and S. Mahmood, "Exploring software security approaches in software development lifecycle: A systematic mapping study," *Comput. Standards Interfaces*, vol. 50, pp. 107–115, Feb. 2017.

[25] N. R. Mead and G. McGraw, "A portal for software security," *IEEE Secur. Privacy Mag.*, vol. 3, no. 4, pp. 75–79, Jul. 2005.

[26] G. Hoglund, and G. McGraw, *Exploiting Software*. Boston, MA, USA: Addison-Wesley, 2004, pp. 1–44.

[27] D. Verdon and G. McGraw, "Risk analysis in software design," *IEEE Secur. Privacy Mag.*, vol. 2, no. 4, pp. 79–84, Jul. 2004.

[28] 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," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5333–5345, Dec. 2016.

[29] Y. Mufti, M. Niazi, M. Alshayeb, and S. Mahmood, "A readiness model for security requirements engineering," *IEEE Access*, vol. 6, pp. 28611–28631, 2018.

[30] M. Ammar, G. Russello, and B. Crispo, "Internet of Things: A survey on the security of IoT frameworks," *J. Inf. Secur. Appl.*, vol. 38, pp. 8–27, Feb. 2018.

[31] X. Hu, Y. Zhuang, Z. Cao, T. Ye, and M. Li, "Modeling and validation for embedded software confidentiality and integrity," in *Proc. 12th Int. Conf. Intell. Syst. Knowl. Eng. (ISKE)*, Nanjing, China, Nov. 2017, pp. 1–6.

[32] M. Khari, Vaishali, and P. Kumar, "Embedding security in software development life cycle (SDLC)," in *Proc. Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, 2016, pp. 2182–2186. [Online]. Available: https://ieeexplore.ieee.org/document/7724651

[33] H. Xu, J. Heijmans, and J. Visser, "A practical model for rating software security," in *Proc. IEEE 7th Int. Conf. Softw. Secur. Rel. Companion*, Jun. 2013, pp. 231–232.

[34] B. B. Mayvan, A. Rasoolzadegan, and Z. G. Yazdi, "The state of the art on design patterns: A systematic mapping of the literature," *J. Syst. Softw.*, vol. 125, pp. 93–118, Mar. 2017.

[35] S. P. Kadam and S. Joshi, "Secure by design approach to improve security of object oriented software," in *Proc. INDIACom*, Mar. 2015, pp. 24–30.

[36] D. Xu and K. E. Nygard, "Threat-driven modeling and verification of secure software using aspect-oriented Petri nets," *IEEE Trans. Softw. Eng.*, vol. 32, no. 4, pp. 265–278, Apr. 2006.

[37] R. Kumar, S. A. Khan, and R. A. Khan, "Analytical network process for software security: A design perspective," *CSI Trans. ICT*, vol. 4, nos. 2–4, pp. 255–258, Dec. 2016.

[38] S. Janakiraman, K. Thenmozhi, J. B. B. Rayappan, and R. Amirtharajan, "Lightweight chaotic image encryption algorithm for real-time embedded system: Implementation and analysis on 32-bit microcontroller," *Microprocessors Microsyst.*, vol. 56, pp. 1–12, Feb. 2018.

[39] T. Diamantopoulos, K. Thomopoulos, and A. Symeonidis, "QualBoa: Reusability-aware recommendations of source code components," in *Proc. MSR*, 2016, pp. 488–491.
[40] Y.-H. Tung, S.-C. Lo, J.-F. Shih, and H.-F. Lin, "An integrated security testing framework for secure software development life cycle," in *Proc. 18th Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Oct. 2016, pp. 1–4.
[41] B. S. Clegg, J. M. Rojas, and G. Fraser, "Teaching software testing concepts using a mutation testing game," in *Proc. ICSE-SEET*, 2017, pp. 33–36.
[42] M. Felderer and F. Elizabeta, "A systematic classification of security regression testing approaches," *Int. J. Softw. Tools Technol. Transf.*, vol. 17, no. 3, pp. 305–319, Jun. 2015.
[43] B. Potter and G. McGraw, "Software security testing," *IEEE Secur. Privacy*, vol. 2, no. 5, pp. 81–85, Sep. 2004.
[44] J.-P. Arcangeli, R. Boujbel, and S. Leriche, "Automatic deployment of distributed software systems: Definitions and state of the art," *J. Syst. Softw.*, vol. 103, pp. 198–218, May 2015.
[45] R. Colomo-Palacios, E. Fernandes, P. Soto-Acosta, and X. Larrucea, "A case analysis of enabling continuous software deployment through knowledge management," *Int. J. Inf. Manage.*, vol. 40, pp. 186–189, Jun. 2018.
[46] S. Velmourougan, P. Dhavachelvan, R. Baskaran, and B. Ravikumar, "Software development life cycle model to improve maintainability of software applications," in *Proc. 4th Int. Conf. Adv. Comput. Commun.*, Aug. 2014, pp. 270–273.
[47] V. Midha and A. Bhattacherjee, "Governance practices and software maintenance: A study of open source projects," *Decis. Support Syst.*, vol. 54, no. 1, pp. 23–32, Dec. 2012.
[48] V. A. Uzunov, B. E. Fernández, and F. Katrina, "Engineering security into distributed systems: A survey of methodologies," *J. Universal Comput. Sci.*, vol. 18, no. 20, pp. 2920–3006, 2012.
[49] Z. Aslanyan, F. Nielson, and D. Parker, "Quantitative verification and synthesis of attack-defence scenarios," in *Proc. IEEE 29th Comput. Secur. Found. Symp. (CSF)*, Lisbon, Portugal, Jun. 2016, pp. 105–119.
[50] T. H. Nguyen, J. Grundy, and M. Almorsy, "Integrating goal-oriented and use case-based requirements engineering: The missing link," in *Proc. ACM/IEEE 18th Int. Conf. Model Driven Eng. Lang. Syst. (MODELS)*, Ottawa, ON, Canada, Sep. 2015, pp. 328–337.
[51] V. G. Vassilakis, H. Mouratidis, E. Panaousis, I. D. Moscholios, and M. D. Logothetis, "Security requirements modelling for virtualized 5G small cell networks," in *Proc. 24th Int. Conf. Telecommun. (ICT)*, Limassol, Cyprus, May 2017, pp. 337–362.
[52] F.-B. Eduardo, *Security Patterns in Practice: Designing Secure Architectures Using Software Patterns*, 1st ed. Hoboken, NJ, USA: Wiley, 2013.
[53] S. Lipner and H. Michael. *The Trustworthy Computing Security Development Lifecycle*. Accessed: Mar. 3, 2019. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms995349.aspx
[54] D. Mellado, E. Fernandez-Medina, and M. A. Piattini, "Common criteria based security requirements engineering process for the development of secure information systems," *Comput. Standards Interfaces*, vol. 29, no. 2, pp. 244–253, 2007.
[55] A. Apvrille and M. Pourzandi, "Secure software development by example," *IEEE Secur. Privacy Mag.*, vol. 3, no. 4, pp. 10–17, Jul. 2005.
[56] J. D. M. Charles, B. Haley, and B. Nuseibeh, "Core security requirements artifacts," Dept. Comput., Fac. Math. Comput., Open Univ., Milton Keynes, U.K., Tech. Rep. 2004/23, 2004. [Online]. Available: http://computing.open.ac.uk
[57] G. Dan. *Build Security in Introduction to the CLASP Process*. [Online]. Available: http://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/requirements/548.html
[58] C. B. Haley, R. Laney, J. Moffett, and B. Nuseibeh, "Security requirements engineering: A framework for representation and analysis," *IEEE Trans. Softw. Eng.*, vol. 34, no. 1, pp. 133–152, Jan./Feb. 2008.
[59] N. R. Mead, E. D. Hough, and T. R. Stehney, II, "Security quality requirements engineering (SQUARE) methodology," Softw. Eng. Inst., Carnegie Mellon Univ., Tech. Rep. CMU/SEI-2005-TR-009, 2005.
[60] J. Manico, *OWASP*, Application Security Verification Standard 3.0.1, 2016, pp. 1–70.
[61] *Information Technology—Security Techniques—Information Security Management Systems—Requirements*, document ISO/IEC 27001:2005, International Organization for Standardization, 2005, pp. 1–34.
[62] C. Thompson, M. Shelton, E. Stark, M. Walker, E. Schechter, and A. P. Felt, "The Web's identity crisis: Understanding the effectiveness of website identity indicators," in *Proc. USENIX*, 2019, pp. 1715–1731.
[63] P. Silva, R. Noël, M. Gallego, S. Matalonga, and A. Hernan, "Software development initiatives to identify and mitigate security threats—A systematic mapping," in *Proc. CibSE*, 2016, pp. 1–15.
[64] A. S. Guinea, G. Nain, and Y. Le Traon, "A systematic review on the engineering of software for ubiquitous systems," *J. Syst. Softw.*, vol. 118, pp. 251–276, Aug. 2016.
[65] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Inf. Softw. Technol.*, vol. 64, pp. 1–18, Aug. 2015.
[66] A. K. Barbara, B. D. Budgen, and O. P. Brereton, "Using mapping studies as the basis for further research—A participant-observer case study," *Inf. Softw. Technol.*, vol. 53, no. 6, pp. 638–651, 2011.
[67] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering—A systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, 2009.
[68] J. Morán, C. Riva, and J. Tuya, "Testing MapReduce programs: A systematic mapping study," *J. Softw. Evol. Process*, vol. 31, no. 3, p. e2120, Mar. 2019.
[69] R. E. Lopez-Herrejon, S. Illescas, and A. Egyed, "A systematic mapping study of information visualization for software product line engineering," *J. Softw. Evol. Process*, vol. 30, no. 2, p. e1912, Feb. 2018.
[70] M. Felderer and J. Carver, "Guidelines for systematic mapping studies in security engineering," in *Empirical Research for Software Security: Foundations and Experience*. Boca Raton, FL, USA: CRC Press, Jan. 2018.
[71] D. Budgen, M. Turner, P. Brereton, and B. Kitchenham, "Using mapping studies in software engineering," in *Proc. PPIG*, 2008, pp. 195–204.
[72] S. Y. Chadli, A. Idri, J. N. Ros, J. L. Fernández-Alemán, J. M. C. de Gea, and A. Toval, "Software project management tools in global software development: A systematic mapping study," *SpringerPlus*, vol. 5, no. 1, p. 2006, Nov. 2016.
[73] M. El Bajta, A. Idri, J. Nicolás, J. Fernández-Alemán, and A. Toval, "Software project management approaches for global software development: A systematic mapping study," *Tsinghua Sci. Technol.*, vol. 23, no. 6, pp. 690–714, Dec. 2018.
[74] C. Wang, M. Daneva, M. Sinderen, and P. Liang, "A systematic mapping study on crowdsourced requirements engineering using user feedback," *J. Softw. Evol. Process*, vol. 31, no. 10, p. e2199, Oct. 2019.
[75] A. Manjavacas, A. Vizcaíno, F. Ruiz, and M. Piattini, "Global software development governance: Challenges and solutions," *J. Softw. Evol. Process*, vol. 32, no. 10, p. e226, Oct. 2020.
[76] D. Šmite, C. Wohlin, T. Gorschek, and R. Feldt, "Empirical evidence in global software engineering: A systematic review," *Empirical Softw. Eng.*, vol. 15, no. 1, pp. 91–118, Feb. 2010.
[77] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *J. Syst. Softw.*, vol. 80, no. 4, pp. 571–583, Apr. 2007.
[78] M. S. Lund, B. Solhaug, and K. Stolen, *Model-Driven Risk Analysis: The CORAS Approach*. Berlin, Germany: Springer, 2011, pp. 1–79.
[79] I. Flechais, M. A. Sasse, and S. M. V. Hailes, "Bringing security home: A process for developing secure and usable systems," in *Proc. NSPW*, 2003, pp. 49–57.
[80] N. R. Mead, *Identifying Security Requirements Using the Security Quality Requirements Engineering (SQUARE) Method* (Integrating Security and Software Engineering). Pittsburgh, PA, USA: Carnegie Mellon Univ., Dec. 2005, pp. 1–21, doi: 10.4018/9781599041476.ch003.
[81] V. Maheshwari and M. Prasana, "Integrating risk assessment and threat modeling within SDLC process," in *Proc. ICICT*, 2016, pp. 1–5.
[82] S. Moyo and E. Mnkandla, "A novel lightweight solo software development methodology with optimum security practices," *IEEE Access*, vol. 8, pp. 33735–33747, 2020.
[83] S. U. R. Khan, I. U. Rehman, and S. U. R. Malik, "The impact of test case reduction and prioritization on software testing effectiveness," in *Proc. Int. Conf. Emerg. Technol.*, 2009, pp. 416–421.
[84] R. Kumar, S. K. Pandey, and S. I. Ahson, "Security in coding phase of SDLC," in *Proc. 3rd Int. Conf. Wireless Commun. Sensor Netw.*, 2007, pp. 118–120.
[85] R. C. Seacord, *Secure Coding in C and C++*. Reading, MA, USA: Addison-Wesley, 2013.
[86] A. Mousa, M. Karabatak, and T. Mustafa, "Database security threats and challenges," in *Proc. ISDFS*, 2020, pp. 1–5.
[87] N. R. Mead, "Measuring the software security requirements engineering process," in *Proc. IEEE 36th Annu. Comput. Softw. Appl. Conf. Workshops*, Jul. 2012, pp. 583–588.

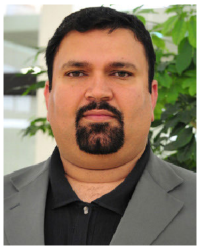
[88] K. Song-Kyoo, "Design of enhanced software protection architecture by using theory of inventive problem solving," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage.*, Dec. 2009, pp. 978–982.
[89] *BSIMM: Building Security In Maturity Model*. Accessed: Mar. 5, 2019. [Online]. Available: https://www.bsimm.com/
[90] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Eng.*, vol. 10, no. 1, pp. 34–44, Jan. 2005.
[91] H. Mouratidis, P. Giorgini, and G. Manson, "When security meets software engineering: A case of modeling secure information systems," *J. Inf. Syst.*, vol. 30, no. 8, pp. 609–629, 2005.
[92] D. Basin, J. Doser, and T. Lodderstedt, "Model driven security: From UML models to access control infrastructures," *ACM Trans. Softw. Eng. Methodol.*, vol. 15, no. 1, pp. 39–91, 2006.
[93] S. Lipner, "The trustworthy computing security development lifecycle," in *Proc. 20th Annu. Comput. Secur. Appl. Conf. (ACSAC)*, 2004, pp. 1–12.
[94] A. van Lamsweerde and E. Letier, "Handling obstacles in goal-oriented requirements engineering," *IEEE Trans. Softw. Eng.*, vol. 26, no. 10, pp. 978–1005, 2000.
[95] M. Alam, J.-P. Seifert, and X. Zhang, "A model-driven framework for trusted computing based systems," in *Proc. EDOC*, 2007, p. 75.
[96] N. R. Mead and T. Stehney, "Security quality requirements engineering (SQUARE) methodology," *ACM SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–7, Jul. 2005.
[97] T. Lodderstedt, D. Basin, and J. Doser, "SecureUML: A UML-based modeling language for model-driven security," in *Proc. Int. Conf. Unified Modeling Lang.*, 2002 pp. 426–441.
[98] M. S. Lund, B. Solhaug, and K. Stølen, *Model-Driven Risk Analysis—The CORAS Approach*. Berlin, Germany: Springer, 2011, pp. 1–358.
[99] 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, no. 1, pp. 12–26, 2007.
[100] C. J. Alberts, A. J. Dorofee, J. F. Stevens, and C. Woody, *Introduction to the OCTAVE Approach*. Pittsburgh, PA, USA: Carnegie Mellon Univ., Software Engineering Institute, 2003, pp. 1–37.
[101] D. P. Gilliam, T. L. Wolfe, J. S. Sherif, and M. Bishop, "Software security checklist for the software life cycle," in *Proc. WET ICE*, 2003, pp. 243–248.
[102] W. Ren, O. Ma, H. Ji, and X. Liu, "Human posture recognition using a hybrid of fuzzy logic and machine learning approaches," *IEEE Access*, vol. 8, pp. 135628–135639, 2020.
[103] A. Agrawal, M. Alenezi, R. Kumar, and R. A. Khan, "Measuring the sustainable-security of Web applications through a fuzzy-based integrated approach of AHP and TOPSIS," *IEEE Access*, vol. 7, pp. 153936–153951, 2019.
[104] F. A. Al-Zahrani, "Evaluating the usable-security of healthcare software through unified technique of fuzzy logic, ANP and TOPSIS," *IEEE Access*, vol. 8, pp. 109905–109916, 2020.
[105] R. Kumar, A. I. Khan, Y. B. Abushark, M. M. Alam, A. Agrawal, and R. A. Khan, "An integrated approach of fuzzy logic, AHP and TOPSIS for estimating usable-security of Web applications," *IEEE Access*, vol. 8, pp. 50944–50957, 2020.
[106] R. Kumar, A. I. Khan, Y. B. Abushark, M. M. Alam, A. Agrawal, and R. A. Khan, "A knowledge-based integrated system of hesitant fuzzy set, AHP and TOPSIS for evaluating security-durability of Web applications," *IEEE Access*, vol. 8, pp. 48870–48885, 2020.
[107] M. Alenezi, A. Agrawal, R. Kumar, and R. A. Khan, "Evaluating performance of Web application security through a fuzzy based hybrid multi-criteria decision-making approach: Design tactics perspective," *IEEE Access*, vol. 8, pp. 25543–25556, 2020.
[108] R. Kumar, A. Baz, H. Alhakami, W. Alhakami, M. Baz, A. Agrawal, and R. A. Khan, "A hybrid model of hesitant fuzzy decision-making analysis for estimating usable-security of software," *IEEE Access*, vol. 8, pp. 72694–72712, 2020.
[109] A. K. Pandey, A. I. Khan, Y. B. Abushark, M. M. Alam, A. Agrawal, R. Kumar, and R. A. Khan, "Key issues in healthcare data integrity: Analysis and recommendations," *IEEE Access*, vol. 8, pp. 40612–40628, 2020.
[110] M. Zarour, M. T. J. Ansari, M. Alenezi, A. K. Sarkar, M. Faizan, A. Agrawal, R. Kumar, and R. A. Khan, "Evaluating the impact of blockchain models for secure and trustworthy electronic healthcare records," *IEEE Access*, vol. 8, pp. 157959–157973, 2020.
[111] A. Agrawal, A. K. Pandey, A. Baz, H. Alhakami, W. Alhakami, R. Kumar, and R. A. Khan, "Evaluating the security impact of healthcare Web applications through fuzzy based hybrid approach of multi-criteria decision-making analysis," *IEEE Access*, vol. 8, pp. 135770–135783, 2020.

RAFIQ AHMAD KHAN received the M.Phil. degree in computer science with a specialization in software engineering from the University of Malakand, Pakistan, under the research supervision of Dr. S. U. Khan, where he is currently pursuing the Ph.D. degree.

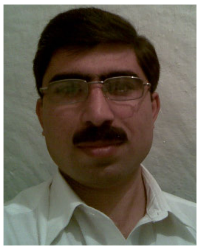
His research interests include software security, empirical software engineering, systematic literature review, requirements engineering, green computing, software testing, agile software development, and global software engineering. He has authored several papers in well-reputed international conferences and journals, including ICGSE and IEEE Access.

SIFFAT ULLAH KHAN received the Ph.D. degree in computer science from Keele University, U.K., in 2011.

He was the Head of the Department of Software Engineering, University of Malakand, Pakistan, for three years, where he was also the Chairman of the Department of Computer Science and IT and is currently an Associate Professor in Computer Science. He is also the Founder and the Leader of the Software Engineering Research Group, University of Malakand. He has successfully supervised ten M.Phil. and four Ph.D. scholars. He has authored over 100 papers, so far, in well-reputed international conferences and journals. His research interests include software outsourcing, empirical software engineering, agile software development, systematic literature review, software metrics, cloud computing, requirements engineering, and green computing/IT. He received the Gold Medal (Dr. M. N. Azam Prize 2015) from the Pakistan Academy of Sciences in recognition of his research achievements in the field of computer (software).



HABIB ULLAH KHAN received the Ph.D. degree in management information systems from Leeds Beckett University, U.K. He is an Associate Professor of MIS with the Department of Accounting and Information Systems, College of Business and Economics, Qatar University, Qatar. He has nearly 20 years of industry, teaching, and research experience. His research interests include IT adoption, social media, Internet addiction, mobile commerce, computer mediated communication, IT outsourcing, big data, and IT security.



MUHAMMAD ILYAS received the Ph.D. degree in computer science from the University of Malakand, Pakistan, where he is currently an Assistant Professor with the Computer Science and IT Department. His research interests include software outsourcing, empirical software engineering, systematic literature review, cloud computing, requirements engineering, and green computing/IT.