International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 6, December 2020, pp. 6507~6520
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i6.pp6507-6520

Ontology-based context-sensitive software security knowledge management modeling

Mamdouh Alenezi
College of Computer and Information Sciences, Prince Sultan University, Saudi Arabia

Article Info

Article history:

Received Mar 3, 2020
Revised May 12, 2020
Accepted May 27, 2020

Keywords:

Fuzzy AHP
Ontology-based context
Security knowledge
Security management
Software security

ABSTRACT

The disconcerting increase in the number of security attacks on software calls for an imminent need for including secure development practices within the software development life cycle. The software security management system has received considerable attention lately and various efforts have been made in this direction. However, security is usually only considered in the early stages of the development of software. Thus, this leads to stating other vulnerabilities from a security perspective. Moreover, despite the abundance of security knowledge available online and in books, the systems that are being developed are seldom sufficiently secure. In this paper, we have highlighted the need for including application context sensitive modeling within a case-based software security management system. Furthermore, we have taken the context-driven and ontology-based frameworks and prioritized their attributes according to their weights which were achieved by using the Fuzzy AHP methodology.

Corresponding Author:

Mamdouh Alenezi,
College of Computer and Information Sciences,
Prince Sultan University,
P.O.Box No. 66833 Rafha Street, Riyadh 11586, Saudi Arabia.
Email: malenezi@psu.edu.sa

1. INTRODUCTION

The digital age has witnessed a large number of businesses being aided and automated by using state-of-the-art web development technologies. E-commerce based applications and their integral contribution to transforming business processes remain an unparalleled success. However, this rising trajectory is beset with an alarming increase in security attacks on such applications [1-2]. The rise in the number of security attacks has led to huge losses for the organizations that are dependent on e-commerce based applications for generating revenue [3]. Security attacks affect the functionality of the application which leads to the unavailability of the service on the internet. This, in turn, has a direct impact on customer satisfaction. Most of the security attacks are experienced as a result of software flaws or vulnerabilities left untended during the software development process. Many Software development processes have not been able to ensure security within the product in the past [4]. Also, the team involved in developing software often lacks the required expertise for generating secure systems.

However, the recent research initiatives have given considerable attention to this lacuna and are working towards security practices that need to be made efficacious during the software development process itself. Software security is a term used to describe security during the whole development procedure of software. To enhance the security of any software, it is imperative to ensure that the software engineers are equipped with the necessary information and mandatory skills for the development of secure software [5]. Only with this elemental knowledge can the software engineers tackle security attacks and deal with security errors in a correct manner. Further, the software engineer's expertise needs to be complemented by security artifacts which assist in understanding the security of the software. To enable the practitioners to gain insight

into the security of the software, there is a need for an automated system that manages the security knowledge and depending on the cases, presents recommendations to the software engineer.

To cite a pertinent example, SHIELDS project targets constructing a secure software engineering environment which is assisted by the repository of the software security knowledge [6]. With the help of the repository, security models can be shared and stored representing the expertise of the specialists. The project provides a modeling tool but lacks the relationship between artifacts and knowledge of software security. Hence, the authors in [7] proposed a management system that manages knowledge and artifacts of software security generated during the development process. The system assists practitioners who may not have the requisite expertise by helping them to analyze heterogeneous cases of software security. However, the work lacks application context-related cases. Modeling software security knowledge in a context-sensitive manner using ontologies can be found in [8] where software security-related knowledge is extracted by assessing the application context at hand.

Anticipating the need for inclusion of application context sensitivity within the case-based management systems, as in [9], is the most efficacious solution. The authors of this paper propose a context-sensitive case-based software security management system. Further, this work prioritizes the artifacts involved in decision making by practitioners for security management. This study is categorized as follows: The second segment on Literature Review discusses the related and relevant work done in this domain. The third segment highlights the need for and significance of the proposed ideation. The segments thereafter discuss the implementations and conclusion.

Literature Review. With the help of semantic tools to assist the security of software, several efforts have been made to achieve ontology-based modeling. Ontologies have clear and formal specifications [6]. Also, ontology is recognized universally as a tool for the modeling of context information. Ontology is being used to provide application context related to security information as in [7, 10]. Some of the pertinent work has been discussed in Table 1.

Table 1. Ontology related pertinent work in security perspective

Year	Title	Summary of Contribution
2016 [3]	Analytical Network Process for Software Security: A Design Perspective	This work presented a novel ontology with a focus on secure web applications. This model was based on SecEval model which was a domain model for describing tailored knowledge objects. Authors integrated the proposed model with UML based web engineering approach and attained good results.
2014 [4]	Risk management perspective in SDLC	Authors in this work produced a new picture of security knowledge artifact which is aimed to assure the requirements of practitioners. This artifact is named Domain Security Met model. This artifact contains knowledge about every security aspect specific to a domain. The use of Domain Security Met was completed on the SecFutur project and results were found to be satisfactory.
2015 [5]	A Case-based Management System for Secure Software Development Using Software Security Knowledge	This work presents a framework for generic Ontology-based user modeling. Also, this work discusses selected inferences of ontology-based user modeling from a different perspective including semantic-enhanced knowledge management and personal knowledge management.
2018 [6]	An Ontology-Based Context Model for Managing Security Knowledge in Software Development	In this paper, the authors have identified the problems associated with necessities on the knowledge desired to make an ICS security assessment. After the problem definition, ICS security knowledge and development life cycle framework for security assessment is developed.
2002 [7]	Knowledge management in software engineering	The study proposed that security knowledge must first integrate features that state what contextual features are to be controlled and signify the knowledge of security in a layout. Further, the layout is logical and satisfactory for the practitioners. Hence, the work proposed to achieve ontology with the context-based approach.

Literature review of the research work and articles in the area of software security, knowledge management, and ontological approaches have paved the ideas for combining and analyzing three of these with a focused temperament on software security. The ontology-based approach is easy to implement by the developers in the security of software. Also, the review has revealed the fact that knowledge management for developers is the prime necessity nowadays where knowledge is everywhere, but it remains unorganized.

2. PROPOSED METHOD

2.1. Needs and significance

In 2006, authors determined that the most significant resources for context modeling are found in the ontology-based models [11]. The study listed six criteria that would be best for context modeling and these six were: richness and quality of information, distributed composition, level of formality, incompleteness,

and ambiguity, partial validation and applicability to existing environments [12-15]. In addition, the study analyzed the markup scheme, key-value, logic-based, graphical, and object-oriented models. The interrelation between software security management and context-driven ontologies has been shown in Figure 1.

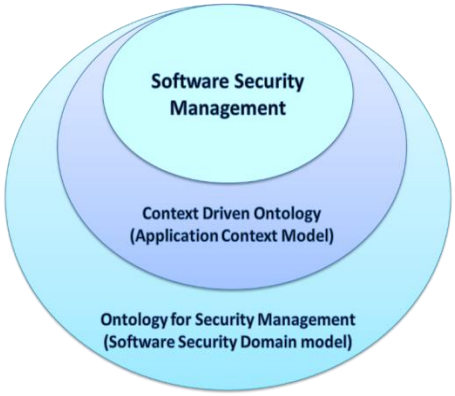

Figure 1. Interrelation between software security management and context-driven ontologies

Figure 1 shows the relationship between software security management, context-driven ontology, and ontology for security management. The concept of ontology plays an important role in the semantic web and particularly in universal computing and next-generation mobile communication systems [16-20]. Ontology can provide a better way of creating associations. It creates real-life scenarios into machine understandable relationships. Further, a context-driven modeling approach for security management also needs a framework that is based on ontology. It will help in diversifying the interrelationships of artifacts depending on security management. The data thus coming from varied sets of information foundations leads to improved user experience.

The problem of security management is also due to the extensive knowledge available on web-based resources which most of the developers use for gaining their knowledge for security services. Hence, an ontology-based and context-sensitive software security management framework would facilitate in gaining an accurate approach for the software developers. This immense challenge needs the specific usage of the tools of ontology and languages which have been introduced in the next section of this paper. Formalizing attributes related to context-driven security modeling and ontology security management criteria to conform the heterogeneity, vagueness, and some quality-related issues. After the critical analysis of the available literature, the authors came up with the two important models of ontology-based context model, which are: Software security domain model and the Application context model. Hierarchy has been shown in Figure 2 and indicated in the ensuing section:

2.1.1. Software security domain model

The ontology-based context model consists of two types. One of them is the software security domain model. The software security domain model is designed with the consideration of the central idea of reviewing important security knowledge resources and is also concerned with the security knowledge repositories such as CWE, stack overflow open question-answer platform, OWASP checklists, and SEI CERT coding guideline, etc. [21, 22]. After this analysis, we divided this analysis further into four security development phases. Elucidation of the major terms used in our ontology is as follows:

a. Security requirement

Designing secure software depends on the security requirements which set a premise for the security guidelines for the developers [23, 24]. Developers need support in deciding the security requirements which further plays a decisive role in the context-based ontology security model.

b. Production practices

Practices that involve designing and coding of a system are termed as production practices and these include design and coding practices [23]. Design practices of security represent practices approved in the system design time. Adopting security design practices may reduce the security risk associated with the production phase. Coding Practices represent a set of rules that are adopted at the code level. Knowledge and context of both levels affect the overall ontology-based context modeling.

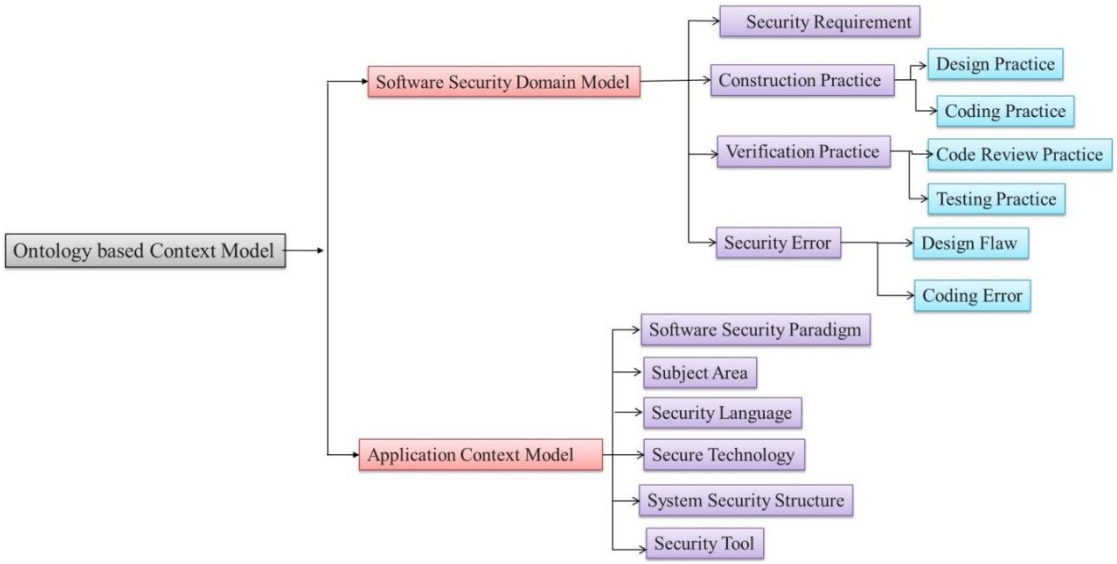

Figure 2. Hierarchical structure of ontology-based context model

c. Verification/validation practices

Verification and Validation ensure that the developed product satisfies the given requirements and that the right product has been developed. These practices include two major processes which are code review and testing process. Description of both is given below:

- Code Review Practice: This practice focuses on identifying security mistakes by the inspection of software at the source code level with the help of different tools such as manual code analysis. This practice also helps to ensure the strengthening of verification and validation practices and, hence, seems important for the building of an ontological based context model [23].
- Testing Practice: This practice focuses on the testing of software while executed in order to find security problems and errors. Most of the errors and problems are found in this level of testing. Hence, it is significant to deliberate it in the preparation of the ontology-based context model [22].
- Security Error: Security error is a noticeable fault during the development of software that may become the cause of a future software weakness [12]. In our ontology, a software security error can be:
- Design Flaw: Design flaw is an unsuitable logical judgment at the design level. A flaw can be instantiated in code but can be a result of a mistake at the design level. These flaws can create major bugs in the future. Hence, looking over these flaws is as important as the manual review of the code [12].
- Coding Error: A code error or a mistake (bug) occurs at the code level. Code error can change the results that were expected to be something else. The fault of the systems is created by a number of coding errors [24].

Both the design flaws and coding errors play a significant role in creating a big security error which further may harm the ontology-based context model.

d. Application context model

The knowledge and application of software security are essential to be put in a framework to develop a context-based ontological model. In our study, we are describing the different attributes that take part in deciding the application in software security for its context. Capturing this context is significant during the process of ontology modeling where context representation depends on the features and relationships created between them. The features are described as follows:

- Software Security Paradigm: The software security paradigm represents the groups of software applications that share some common characteristics. Security paradigm refers to where all the security engineering concepts pertaining to the development of security are applied. For example, Web application security, desktop application security, mobile security, etc., [17].
- Subject Area: It signifies domains that a security application belongs to. For instance, Banking, Defense systems, health, Travel, etc. It signifies the vital elements of the security attributes of the software. The security feature is related to the software as well [25].
- Security Language: It signifies the programming language used to improve a secure application. For example, Java, JavaScript, and other high-level security languages [26].

- Secure Technology: It represents a collection of security tools and frameworks that are used along with programming languages to develop security, for example, Web security framework toolkit, SDK, OWASP guidelines [27].
- System Security Structure: It contains the secure structure in which the application has to be implemented. For example, Secure Database management system and other run time platforms
- Security Tool: Security tools consist of the concrete structure that is implemented towards the specification of security in the application. For example, HTML Purifier [25].

Figure 2 shows the complete hierarchical structure including the interrelationships of the software security domain model and the application context model. Authors tried to create hierarchical relationships between both of these sub-attributes. Software security domain model and its attributes contain specific phases of security development such as security requirement, construction practices, verification, and validation practice and security error which further depend on their sub-attributes which are design practice, coding practice, code review practice, testing practice, design flaw, and coding error. Application context modeling contains artifacts such as software security paradigm, subject area, security language, secure technology, system security structure, and security tools.

The hierarchical structure of the ontology-based context model shows that different artifacts and factors decide the modeling of the context model. But their contribution to modeling is not known. To know the different contributions of each artifact, a qualitative analysis of the ontology-based context model is to be done.

2.1.2. Evaluation criteria

a. Context-driven security modeling criteria

Model-driven or context-driven security is a contemporary topic for which the software developers are being asked to carry out security tests. But, quite often, security developers confront the dilemma of where to start and where to end this and in which context should they start their test. Context-driven security modeling is an apt solution for such questions and ambiguities [27]. The criteria on which the security-based context modeling should be done are also the reasons for this confusion. In this research, the authors are focusing on the criteria with their defined priority to ease the problems of developers. Table 2 shows the different criteria on which the context-driven modeling should be done.

Thus, the non-deterministic contextual information is what is available at any point in time. The ontologies and the value ranges cleared herein provide means to address these issues by confining the unpredictability of contextual data. Figure 3 shows the interrelationships between the artifacts of context driven security modeling.

Table 2. Context-driven security modeling criteria

Usability	The usability of the software or application is the first which is affected while ensuring security. For this reason, researchers usually call security and usability two different sides of a coin. Hence, ensuring both is a challenge and priority as well [28]. Usability is termed as the ease of use and learnability of software. The degree of usability defines how easy it is going to be for the end-user to handle the system.
Quality	The quality of the application system is well affected by its security. Ensuring quality increases the reliability of the user to the system, as it believes that the specified requirements are fulfilled. For this reason, quality becomes an important and considerable artifact of context-driven security modeling [26].
Applicability	A model is developed for a specific reason and its applicability for that reason should be higher. This attribute considers the usability and applicability of the context model within existing infrastructures [25].
Comparability	Different applications of the same system give different results. Hence, it is essential to deliver a means to compare values including different units and encodings, etc., Thus, the comparability of the model should be considered while designing it [28].
Traceability	To provide adequate information about the context and origin, the formulations of tools should be known to the developer. Here, the traceability of the system becomes important in the context-based ontological system [26].
Acceptability	Acceptability deals with the accordance or agreement of measured or derived information with the well-defined context model. A model should define the range that a context value can take, or define a particular co-existence of values to be impossible [29].
Inference	Inference can be defined as the conclusions drawn by evidence collected. In context model terms, the process of making context information is openly available from other context sources [30].

b. Ontology security management criteria

The second set of criteria is used to assess the ontologies of security management including flexibility, extensibility, and completeness of the ontology, consistency, and granularity of the concepts and properties, as well as the flexibility applied. The description of each artifact is given in Table 3. The growing dependency on secure systems preserves the need for ontology development of security management. Ensuring the consistency of ontology developed for security management is important and largely depends

on its artifacts, which are defined in Table 3. Although every artifact contributes to the production of a better ontology for security management, still there are some artifacts that should be given preference over others. Hence to quantify the preference of artifacts, the authors propose a methodology followed with Fuzzy AHP to quantify the priority of ontology security management artifacts. Figure 4 shows the interrelationships of artifacts in the ontology of security management.

Table 3. Ontology security management criteria

Reusability	Reuse of knowledge and specification process of security requirements during software development is an important concern [31]. Increasing the reusability improves the expansion of using the ontology among many other tasks.
Flexibility	Flexibility is essential in managing policies across multiple domains, flexibility in the level of abstraction, flexibility across different environments, etc. There are multiple scenarios faced in ontology security management that need flexibility. Hence, it appears to be an important cognition in ontology-based security management criteria [32].
Extensibility	Extensibility refers to the possibility of extending new definitions to the ontology without altering the existing dependencies. The strength and new updates that an application can accept can be defined under extensibility [33].
Granularity	Granularity is related to collating different concepts to create a better ontology for security management [34].
Consistency	A consistency check is about testing the existence of obvious or understood flaws in the signified ontological security management model [35].
Completeness	An ontology for security management is said to be complete if it covers the domain for which it is developed. Completeness of ontology depends on its boundaries and limits [36].
Redundancy	This artifact tests for the repetition of logical flows. This is challenging and time-consuming [37].
Readability	Readability can be related to usability and quality as well, but in the ontology of the security management model, readability prefers checking for security policies and guidelines that are being used in security management [38].
Scalability	Scalability refers to determining the scale of ontology which could be large for major applications and limited for small scale applications. The scalability of ontology also defines its boundaries [38].

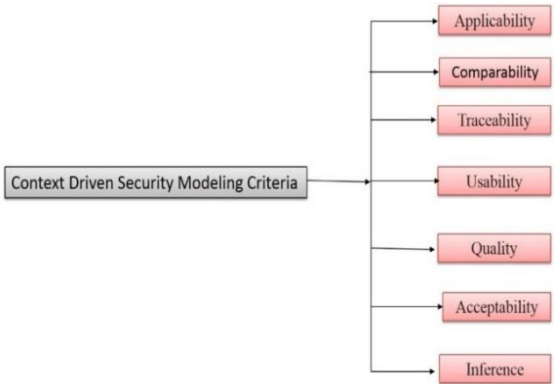

Figure 3. Interrelationship of context-driven security modeling and its artifacts

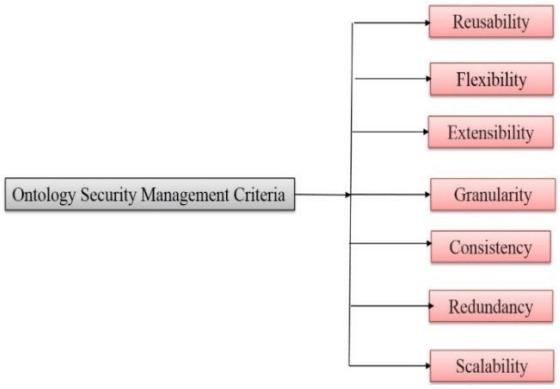

Figure 4. Interrelationship of ontology security management criteria and its artifacts

3. RESEARCH METHOD

Till now we have defined the specific artifacts of ontology-based context model and criteria of ontology-based security management and context-driven security modeling. 10, 7, and 7 attributes were found, respectively, which affect the ontology-based context modeling of security management. Now the pertinent question that arises is that among these numbers of attributes which is a more important concern and which one is not. To solve this issue, the authors came up with prioritizing these attributes according to their weight of contribution towards their respective models. To prioritize the attributes which are in a hierarchical format, authors are using the Fuzzy AHP method for decision making. With the help of Fuzzy AHP, there is a need to assess these attributes of ontology-based context-driven modeling for ensuring the security of software for satisfaction and ease of usage. The multi-criteria problem is decomposed into a hierarchy using AHP, and it was adopted by the author [31]. It is also used to measure the priority and importance of every attribute.

Further, AHP is considered as a better method than every other MCDM method such as ELECTRE. But, still, AHP cannot resolve the uncertainty and vagueness related to the mapping of a decision maker's awareness of exact numbers. To deal with uncertainty and vagueness authors have combined AHP and fuzzy into one. In this work, Fuzzy AHP is chosen for assessing the security of ontology because context-sensitive security management is proficient in handling multiple criteria decision-making problems very easily [33]. It is also capable of converting qualitative or linguistic inputs into quantitative or numerical results.

Further, the results are an effective assessment of security management in the form of weight and ranking [34]. For assessing the ontology-based security model using experts' data and reaching an agreement among the experts, this work implements the Buckley method [32] and also uses the eigenvector method to estimate the weights of attributes. The first step is to create a pair-wise comparison method from expert's opinions because the AHP method only uses the pair-wise comparison matrix to estimate ambiguity in MCDM difficulties. The Fuzzy AHP method contains four major steps which are deliberated below:

The first step is describing triangular fuzzy numbers for the paired linguistic values. A Triangular Fuzzy Number (TFN) is represented as (Lo, Mi, Up). The equations (1-3) are used in changing the linguistic values into TFN [18] and denoted as (Loij, Miij, Upij) where, Loij is lowermost value, Miij is middle value and Upij is uppermost level values assigned to linguistic values. Further, TFN [ηij] is recognized as the succeeding:

$$\eta_{ij} = [Lo_{ij}, Mi_{ij}, Up_{ij}]$$

where $Lo_{ij} \leq Mi_{ij} \leq Up_{ij}$

$$Lo_{ij} = min(J_{ijk}) \quad (1)$$

$$Mi_{ij} = (J_{ij1}, J_{ij2} \ldots\ldots\ldots J_{ijk})^{1/k} \quad (2)$$

$$Up_{ij} = max(J_{ijk}) \quad (3)$$

In the above equations, J_{ijk} is showing the comparative value of ij with reference to expert k, where i and j signify a pair of criteria being judged by practitioners. Value ηij is estimated based on the geometric mean of practitioner's views for a specific judgment. Further, after the construction of pair-wise comparisons a matrix different fuzzy operation is performed on it and then defuzzification is performed. This work used alpha cut method for defuzzification [18] where alpha cut method as formulated in (4)-(6).

$$\mu_{\alpha,\beta}(\eta_{jj}) = [\beta.\eta_{\alpha}(Lo_{ij}) + (1-\beta).\ \eta_{\alpha}(Up_{ij})] \quad (4)$$

where $0 \leq \alpha \leq 1$ *and* $0 \leq \beta \leq 1$. Such that,

$$\eta_{\alpha}(Lo_{ij}) = (Mi_{ij} - Lo_{ij}).\alpha + Lo_{ij} \quad (5)$$

$$\eta_{\alpha}(Up_{ij}) = Up_{ij} - (Up_{ij} - Mi_{ij}).\alpha \quad (6)$$

Where α and β in these equations are used for the preferences of experts and intolerance of experts respectively. The values of α and β vary between 0 and 1. The maximum or threshold value of α is any value taken from a scale of 0 to 1, which has its membership value greater than or equal to an alpha threshold value, represented by α. Crisp sets ρα,β (Ã) simply describe whether an element is either a member of the set or not. The single pair-wise comparison matrix is expressed in (8) [32].

After evaluating a single pair-wise comparison matrix, eigenvectors have to be determined. The next step is to determine the eigenvalue and eigenvector of the pair-wise comparison matrix. To determine the aggregated weight of particular criteria, the eigenvector is calculated.

$$\rho_{\alpha,\beta}(\tilde{A}) = \rho_{\alpha,\beta}[\tilde{a}_{ij}] = \begin{array}{c} \\ C_1 \\ C_2 \\ \vdots \\ C_n \end{array} \begin{array}{c} \begin{array}{ccc} C_1 & C_2 \ldots\ldots\ldots\ldots & C_n \end{array} \\ \begin{bmatrix} 1 & \rho_{\alpha,\beta}(\tilde{a}_{11}) \ldots\ldots & \rho_{\alpha,\beta}(\tilde{a}_{1i}) \\ 1/\rho_{\alpha,\beta}(\tilde{a}_{21}) & 1 \ldots\ldots & \rho_{\alpha,\beta}(\tilde{a}_{2i}) \\ . & . & . \\ . & . & . \\ 1/\rho_{\alpha,\beta}(\tilde{a}_{j1}) & 1/\rho_{\alpha,\beta}(\tilde{a}_{j2}) \ldots\ldots & 1 \end{bmatrix} \end{array} \quad (7)$$

Let us assume that μ is denoting the eigenvector while λ denotes the eigenvalue of fuzzy pair-wise comparison matrix ηij. Then,

$$[\mu_{\alpha,\beta}(\eta_{ij}) - \lambda I].\ \mu = 0 \quad (8)$$

In (8) symbol I signify the unitary matrix. By applying equations (1-8), the weights of every attribute with respect to all other attributes may be attained. For checking the consistency and continuing the AHP process, check the consistency ratio (CR) [31]. If CR value is less than 0.1, the AHP analysis is correct otherwise analyze the AHP process again.

4 RESULTS AND DISCUSSION

For implementing the abovementioned methodology of Fuzzy AHP, we prepared three questionnaires for the ontology-based context model, context-driven security modeling criteria, and ontology security management criteria. These questionnaires were distributed to experts and the profile of experts included developers, researchers, and experts from organizations. 40 valid responses were collected and according to these data and implementing equations (1)-(8) on these data, the authors came up with the results that are as follows:

4.1. Implementation for ontology-based context model

Table 4 represents the combined pair-wise judgment matrix for level 1 of the hierarchal tree. For simplicity, the artifacts have been named as Software Security Domain Model (C1) and Application Context Model (C2). Table 5 represents the combined pair-wise judgment matrix for level 2 attributes. For ease, the attributes have been named as security requirement (C11), Construction practice (C12), Verification practice (C13), and Security error (C14). Table 6 represents the combined pair-wise judgment matrix for level 2 attributes. For ease, the attributes have been named as software paradigm (C21), subject area (C22), Language (C24), Secure Technology (C24), System Structure (C25), and Security tool as C26.

Table 4. Aggregated pair-wise comparison matrix at level 1

	Software Security Domain Model (C1)	Application Context Model (C2)
Software Security Domain Model (C1)	1,1,1	1.0660, 1.5280, 1.9800
Application Context Model (C2)	-	1,1,1

Table 5. Combined pair-wise judgment matrix at level 2 for software security domain model

	Security Requirement (C11)	Construction Practice (C12)	Verification Practice (C13)	Security Error (C14)
Security Requirement (C11)	1,1,1	1.3990, 1.8160, 2.4460	1.6050, 2.3360, 3.1470	1.0850, 1.3430, 1.8720
Construction Practice (C12)	-	1,1,1	0.4810, 0.6070, 0.8530	1.1920, 1.4890, 1.8980
Verification Practice (C13)	-	-	1,1,1	0.1990, 0.2950, 0.4630
Security Error (C14)	-	-	-	1,1,1

Table 6. Combined pair-wise judgment matrix at level 2 for application context model

	Software Security Paradigm (C21)	Subject Area (C22)	Security Language (C23)	Secure Technology (C24)	System Security Structure (C25)	Security Tool (C26)
Software Security Paradigm (C21)	1,1,1	1.0640, 1.5290, 1.9900	0.5110, 0.5980, 0.8590	1.7290, 2.3110, 2.9010	1.6920, 2.4140, 3.1470	1.5760, 2.0930, 2.613
Subject Area (C22)	-	1,1,1	1.1820, 1.4740, 1.8720	0.7910, 0.9600, 1.1350	1.4590, 1.8590, 2.2150	1.3330, 1.5230, 1.7970
Security Language (C23)	-	-	1,1,1	1.0850, 1.3430, 1.8720	1.6050, 2.3360, 3.1470	0.3350, 0.4270, 0.574
Secure Technology (C24)	-	-	-	1,1,1	1.4960, 1.9280, 2.3540	0.9450, 1.0810, 1.6370
System Security Structure (C25)	-	-	-	-	1,1,1	1.1870, 1.5350, 2.0280
Security Tool (C26)	-	-	-	-	-	1,1,1

Table 7 represents the combined pair-wise judgment matrix for construction practice at level 3. Attributes have been named as Design practice (C121) and Coding Practice (C122). Table 8 shows the combined pair-wise comparison matrix for verification practice at level 3. Attributes have been renamed as code review practice (C131) and Testing Practice (C132). Table 9 represents the combined pair-wise judgment matrix for security error at level 3. Attributes have been named as Design flow (C141) and Coding error (C142).

Table 7. Combined pair-wise judgment matrix at level 3 for construction practice

	Design Practice (C121)	Coding Practice (C122)
Design Practice (C121)	1,1,1	1.3750, 1.7180, 2.1780
Coding Practice (C122)	-	1,1,1

Table 8. Combined pair-wise judgment matrix at level 3 for verification practice

	Code Review Practice (C131)	Testing Practice (C132)
Code Review Practice (C131)	1,1,1	0.3350, 0.4270, 0.5740
Testing Practice (C132)	-	1,1,1

Table 9. Combined pair-wise judgment matrix at level 3 for security error

	Design Flaw (C141)	Coding Error (C142)
Design Flaw (C141)	1,1,1	0.9450, 1.0810, 1.6370
Coding Error (C142)	-	1,1,1

Defuzzification is performed using (4)-(8) from the abovementioned methodology and defuzzified matrix of each pair-wise comparison matrix is shown from Table 10 to Table 15. Table 10 shows the defuzzifed matrix of level 1 attributes and local weights have been obtained as C1 is 0.6400 and C2 is 0.3600. Table 11 shows the defuzzifed matrix of level 2 attributes and local weights have been obtained as C11 is 0.3571, C12 is 0.2705, C13 is 0.1840, C14 is0.1884.

Table 10. Defuzzified matrix and local weights for ontology-based context model

	Software Security Domain Model (C1)	Application Context Model (C2)	Local Weights
Software Security Domain Model (C1)	1	1.7780	0.6400
Application Context Model (C2)	0.5624	1	0.3600
	CR= 0.0003		

Table 11. Defuzzified matrix and local weights for software security domain model at level 2

	Security Requirement (C11)	Construction Practice (C12)	Verification Practice (C13)	Security Error (C14)	Local Weights
Security Requirement (C11)	1	1.8640	1.7780	1.4110	0.3571
Construction Practice (C12)	0.5360	1	1.7740	1.6650	0.2705
Verification Practice (C13)	0.5620	0.5640	1	1.1260	0.1840
Security Error (C14)	0.7090	0.6010	0.8880	1	0.1884
		CR= 0.0145			

Table 12. Defuzzified matrix and local weights for application context model at level 2

	Software Security Paradigm (C21)	Subject Area (C22)	Security Language (C23)	Secure Technology (C24)	System Security Structure (C25)	Security Tool (C26)	Local Weights
Software Security Paradigm (C21)	1	1.7780	0.8920	2.5630	2.6670	2.3440	0.2650
Subject Area (C22)	0.5620	1	1.7510	1.2120	1.8530	1.7940	0.1921
Security Language (C23)	1.1210	0.5710	1	0.9890	2.6060	0.6910	0.1678
Secure Technology (C24)	0.3900	0.8250	1.0110	1	2.1770	0.7710	0.1404
System Security Structure (C25)	0.3750	0.5400	0.3840	0.4590	1	1.8210	0.1049
Security Tool (C26)	0.4270	0.5570	1.4470	1.2970	0.5490	1	0.1298
			CR=0.0430				

Table 13. Defuzzified matrix and local weights for construction practice at level 3

	Design Practice (C121)	Coding Practice (C122)	Local Weights
Design Practice (C121)	1	1.9980	0.6664
Coding Practice (C122)	0.5020	1	0.3336
	CR= 0.0006		

Table 14. Defuzzified matrix and local weights for verification practice at level 3

	Code Review Practice (C131)	Testing Practice (C132)	Local Weights
Code Review Practice (C131)	1	0.6910	0.4086
Testing Practice (C132)	1.4472	1	0.5914
	CR=0.0005		

Table 15. Defuzzified matrix and local weights for security error at level 3

	Design Flaw (C141)	Coding Error (C142)	Local Weights
Design Flaw (C141)	1	0.7710	0.4354
Coding Error (C142)	1.2970	1	0.5646
	CR= 0.0008		

After the calculation of local weights, the final weight of each attribute is to be calculated and Table 16 is showing the final weights and with the overall priority being calculated. Figure 5 denotes the graphical notation of the final weights of attributes of the ontology-based context model. It is clear from Figure 5 that the security requirement attribute is the most significant one and system security structure has got the lowest priority amongst all.

Table 16. Overall weights and priorities

First Level Attributes	Local Weights of First Level	Second Level Attributes	Local Weights of Second Level	Final Weights of Second Level	Third Level Attributes	Local Weights of the Third Level	Overall Weights	Overall Priority
C1	0.6400	C11	0.3571	0.2285	-	-	0.2285	1
		C12	0.2705	0.1731	C121	0.6664	0.1154	2
					C122	0.3336	0.0577	8
		C13	0.1840	0.1178	C131	0.4086	0.0481	11
					C132	0.5914	0.0697	4
		C14	0.1884	0.1206	C141	0.4354	0.0525	9
					C142	0.5646	0.0681	5
C2	0.3600	C21	0.2650	0.0954	-	-	0.0954	3
		C22	0.1921	0.0692	-	-	0.0692	6
		C23	0.1678	0.0604	-	-	0.0604	7
		C24	0.1404	0.0505	-	-	0.0505	10
		C25	0.1049	0.0378	-	-	0.0378	13
		C26	0.1298	0.0467	-	-	0.0467	12

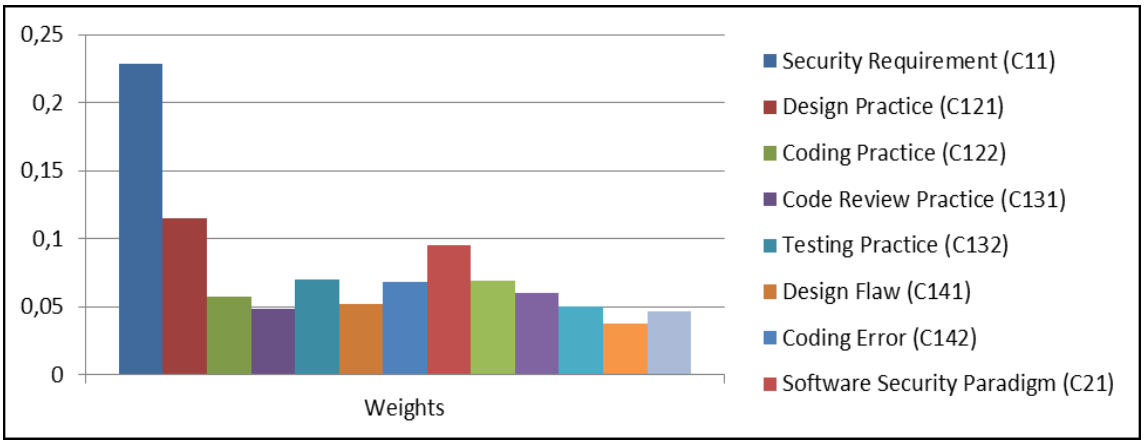

Figure 5. Graphical representation of final weights of ontology-based context model

4.2. Implementation for ontology security management criteria

Table 17 enlists the combined pair-wise comparison matrix for the ontology security management perspective. For the ease of calculation, the artifacts have been named as Applicability (F1), Comparability (F2), Traceability (F3), Usability (F4), Quality (F5), Acceptability (F6) and Inference (F7).

Table 17. Aggregated pair-wise comparison matrix for ontology security management criteria

	Applicability (F1)	Comparability (F2)	Traceability (F3)	Usability (F4)	Quality (F5)	Acceptability (F6)	Inference (F7)
Applicability (F1)	1,1,1	0.5520, 0.6390, 0.9050	1.2870, 1.5230, 2.1080	0.4810, 0.6070, 0.8530	1.1920, 1.4890, 1.8980	0.3980, 0.5110, 0.6620	0.4110, 0.5380, 0.7310
Comparability (F2)	-	1,1,1	1.5530, 2.2000, 2.8500	1.7340, 2.2020, 2.6920	0.7910, 0.9600, 1.1350	1.7340, 2.2020, 2.6920	1.4790, 1.8590, 2.2150
Traceability (F3)	-	-	1,1,1	1.3990, 1.8160, 2.4460	1.2600, 1.8050, 2.2170	0.1990, 0.2950, 0.4630	0.538, 0.8130, 1.2770
Usability (F4)	-	-	-	1,1,1	1.2500, 1.6390, 2.0280	1.2870, 1.5910, 2.0000	0.4720, 0.7060, 1.2520
Quality (F5)	-	-	-	-	1,1,1	1.1920, 1.4890, 1.8980	0.5780, 0.7330, 0.9580
Acceptability (F6)	-	-	-	-	-	1,1,1	0.677, 0.749, 1.027
Inference (F7)	-	-	-	-	-	-	1,1,1

Solving the fuzzified values using (1)-(4) and defuzzying using (4)-(8), we got the defuzzified values in Table 18. Weights with the priority of each attribute are also shown in Table 18. Figure 6 maps the graphical representation of the attributes of ontology security management criteria. It is evident from Figure 6 that the Comparability has the highest priority and acceptability has the lowest priority among all.

Table 18. Defuzzified matrix and weights for ontology security management criteria

	Applicability (F1)	Comparability (F2)	Traceability (F3)	Usability (F4)	Quality (F5)	Acceptability (F6)	Inference (F7)	Weights	Priority
Applicability (F1)	1	0.9340	1.8600	1.7740	1.6650	1.4360	0.8050	0.1761	2
Comparability (F2)	1.0707	1	2.4150	2.4580	1.2120	2.4580	1.8530	0.2261	1
Traceability (F3)	0.5376	0.4141	1	2.1200	2.0220	1.1260	1.1120	0.1438	3
Usability (F4)	0.5637	0.4069	0.4717	1	1.8900	1.0010	1.0340	0.1117	6
Quality (F5)	0.6006	0.8251	0.4946	0.5291	1	1.7670	1.0010	0.1150	5
Acceptability (F6)	0.6964	0.4068	0.8881	0.9990	0.5659	1	1.0510	0.1026	7
Inference (F7)	1.2422	0.5397	0.8993	0.9671	0.9990	0.9515	1	0.1247	4
				CR=0.02457					

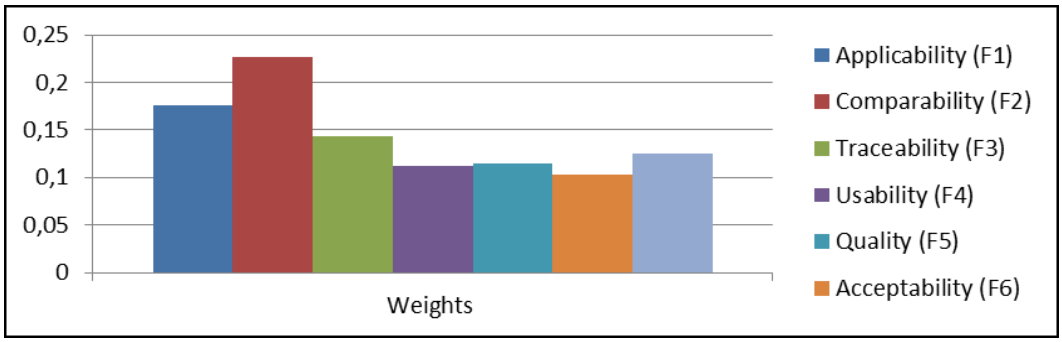

Figure 6. Graphical representation of final weights of ontology security management criteria

4.3. Implementation for context-driven security modeling criteria

Table 19 enunciates the aggregated pair-wise comparison matrix for context-driven security modeling criteria. The attributes have been named as Reusability (A1), Flexibility (A2), Extensibility (A3), Granularity (A4), Consistency (A5), Redundancy (A6) and Scalability (A7).

Table 19. Aggregated pair-wise comparison matrix for context-driven security modeling criteria

	Reusability (A1)	Flexibility (A2)	Extensibility (A3)	Granularity (A4)	Consistency (A5)	Redundancy (A6)	Scalability (A7)
Reusability (A1)	1,1,1	0.6900, 0.8900, 1.1000	0.6601, 1.1700, 1.6900	0.7000, 0.9500, 1.3500	1.1900, 1.5800, 2.1500	0.2300, 0.2800, 0.3600	1.1500, 1.4400, 1.7000
Flexibility (A2)	-	1,1,1	0.3100, 0.3900, 0.5600	0.2300, 0.2800, 0.3600	0.7910, 0.9600, 1.1350	1.7340, 2.2020, 2.6920	1.4790, 1.8590, 2.2150
Extensibility (A3)	-	-	1,1,1	1.1500, 1.4400, 1.7000	0.6900, 0.8900, 1.1000	0.6600, 1.1700, 1.6900	0.7000, 0.9500, 1.3500
Granularity (A4)	-	-	-	1,1,1	1.1900, 1.5800, 2.1500	0.2300, 0.2800, 0.3600	1.1500, 1.4400, 1.7000
Consistency (A5)	-	-	-	-	1,1,1	0.2300, 0.2800, 0.3600	0.7910, 0.9600, 1.1350
Redundancy (A6)	-	-	-	-	-	1,1,1	0.6900, 0.8900, 1.1000
Scalability (A7)	-	-	-	-	-	-	1,1,1

Defuzzification is performed using (4)-(8). The overall weights along with their corresponding priority have been shown in Table 20. Figure 7 depicts the graphical representation of attributes of context-driven security modeling criteria. It can be seen from Figure 7 that redundancy has the highest priority and consistency has the lowest priority among all.

Table 20. Defuzzified matrix and weights for context-driven security modeling criteria

	Reusability (A1)	Flexibility (A2)	Extensibility (A3)	Granularity (A4)	Consistency (A5)	Redundancy (A6)	Scalability (A7)	Weights	Priority
Reusability (A1)	1	0.8900	1.1700	0.9900	1.6300	0.2900	1.3600	0.1168	6
Flexibility (A2)	1.1236	1	0.4100	0.2900	1.2120	2.4580	1.8530	0.1520	4
Extensibility (A3)	0.8547	2.4390	1	1.3600	0.8900	1.1700	0.9900	0.1577	2
Granularity (A4)	1.0101	3.4482	0.7353	1	1.6300	0.2900	1.3600	0.1565	3
Consistency (A5)	0.6135	0.8251	1.1236	0.6135	1	0.2900	1.2120	0.0928	7
Redundancy (A6)	3.4482	0.4068	0.8547	3.4483	3.4483	1	0.8900	0.2175	1
Scalability (A7)	0.7353	0.5370	1.0101	0.7354	0.8251	1.1236	1	0.1067	6

CR=0.03507

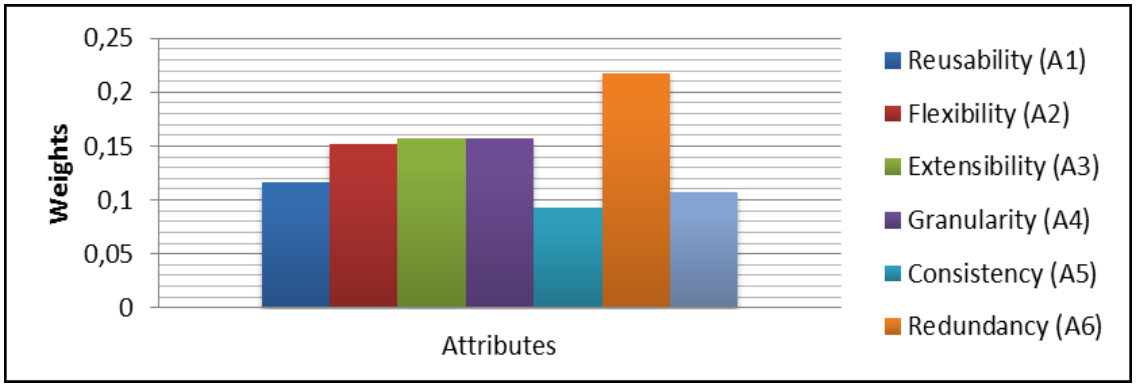

Figure 7. Graphical representation of attributes of context-driven security modeling criteria

4.4. Discussion

This research work is focused on providing help to those developers who have no idea of security knowledge management and who don't have any idea of where to begin and when to stop. The proposed work here has taken three important model frameworks which are: ontology-based context model, ontology security management criteria, and context-driven security modeling criteria. The core intent is to prioritize the attributes or artifacts contributing to these three models. This prioritization is performed using the famous multi-criteria decision-making technique- Fuzzy AHP. This prioritization and ranking help the developers to find the highest priority attribute and make them focus on that particular attribute for managing the knowledge on security guidelines and procedures. According to the results achieved, the following points of discussion that become nodal are:

- Security requirement has the highest priority among all attributes of the ontology-based context model. Hence it might be said that security requirements are responsible for a secure and proven good ontology-based context model.
- Comparability is the highest priority attribute amongst all the attributes of ontology security management criteria. From this, it can be inferred that the comparability of an ontology security management is responsible for its successful implementation. Developers should focus on the comparability of security management while preparing ontology for any software.
- Redundancy is found to be the highest weighted attribute amongst all attributes of context-driven security modeling criteria. For this, the developers should focus on minimizing redundancy to prepare a context-driven model.
- Fuzzy AHP is found to give precise results. Though there has been no comparison made for results, it can be done in the future using other methods of decision making.

5. CONCLUSION

Context-driven ontology for security management is an effective mechanism to analyze the better framework, guidelines, or tools for assuring security. This paper presents a new way of analysis of ontology-based security management modeling using Fuzzy AHP as an analysis mechanism. Furthermore, this work can assist developers in prioritizing their ontology-based framework accordingly and save the time invested in and the cost incurred over software. It also helps in making better choices, since it allows the developers to assist themselves by ranking attributes according to their specification.

REFERENCES

[1] F. Liu, et al., "Unsupervised heterogeneous domain adaptation via shared fuzzy equivalence relations," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 6, pp. 3555-3568, 2018.

[2] A. K. Pandey, et al., "Key Issues in Healthcare Data Integrity: Analysis and Recommendations," *IEEE Access*, vol. 8, pp. 40612-40628, 2020.

[3] R. Kumar, et al., "Analytical Network Process for Software Security: A Design Perspective," *CSI Transactions on ICT*, vol. 4, no. 2, pp. 255-258, 2016.

[4] K. Sahu, Rajshree and R. Kumar, "Risk management perspective in SDLC," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 3, pp. 1247-1251, 2014.

[5] M. Saito, et al., "A Case-based Management System for Secure Software Development Using Software Security Knowledge," *Procedia computer science*, vol. 60, pp. 1092-1100, 2015.

[6] S. F. Wen and B. Katt, "An Ontology-Based Context Model for Managing Security Knowledge in Software Development," *Proceedings of the 23rd Conference of Open Innovations Association FRUCT*, pp. 416-424, 2018.

[7] I. Rus and M. Lindvall, "Knowledge management in software engineering," *IEEE Software*, vol. 19, no. 3, pp. 26-38, 2002.

[8] M. Alenezi and F. I. Khan, "Context-Sensitive Case-Based Software Security Management System," in *Proceedings of the Computational Methods in Systems and Software*, pp. 135-141, 2019.

[9] R. Kumar, et al., "An Integrated Approach of Fuzzy Logic, AHP and TOPSIS for Estimating Usable-Security of Web Applications," *IEEE Access*, vol. 8, pp. 50944-50957, 2020.

[10] J. Xie, et al., "Why do programmers make security errors?" in *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 161-164, 2011.

[11] S. Schaffert, "IkeWiki: A semantic wiki for collaborative knowledge management," in *15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'06)*, pp. 388-396, 2006.

[12] R. Kumar, et al., "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.

[13] A. Agrawal, et al., "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.

[14] T. Takahashi, et al., "Ontological approach toward cybersecurity in cloud computing," in *Proceedings of the 3rd International conference on Security of information and networks*, pp. 100-109, 2010.

[15] L. Razmerita, "An ontology-based framework for modeling user behavior—A case study in knowledge management," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 41, no. 4, pp. 772-783, 2011.
[16] K. Sahu and R. K. Srivastava, "Revisiting Software Reliability," *Data Management, Analytics and Innovation*, pp. 221-235, 2019.
[17] K. Sahu and R. K. Srivastava, "Soft Computing Approach for Prediction of Software Reliability," *ICIC Express Letters*, vol. 12, no. 12, pp. 1213-1222, 2018.
[18] R. Kumar, et al., "A Hybrid Model of Hesitant Fuzzy Decision-Making Analysis for Estimating Usable-Security of Software," *IEEE Access*, vol. 8, no. 4, pp. 72694-72712, 2020.
[19] M. Bishop, "A Clinic for 'Secure' Programming," *IEEE Security & Privacy*, vol. 8, no. 2, pp. 54-56, 2010.
[20] R. Kumar, et al., "Fuzzy-Based Symmetrical Multi-Criteria Decision- Making Procedure for Evaluating the Impact of Harmful Factors of Healthcare Information Security," *Symmetry*, vol. 12, no. 4, pp. 664-686, 2020.
[21] F. I. Khan, et al., "Security assessment of four open source software systems," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 16, no. 2, pp. 860-881, 2019.
[22] M. Alenezi and Y. Javed, "Developer companion: A framework to produce secure web applications," *International Journal of Computer Science and Information Security*, vol. 14, no. 7, pp. 12-16, 2016.
[23] K. Sahu and R. Shree, "Software Security: A Risk Taxonomy," *International Journal of Computer Science & Engineering Technology*, vol. 6, no. 2, pp. 36-41, 2015.
[24] M. T. J. Ansari, et al., "Store: Security threat oriented requirements engineering methodology," *Journal of King Saud University-Computer and Information Sciences*, 2018.
[25] R. Kumar, et al., "Measuring the Security Attributes through Fuzzy Analytic Hierarchy Process: Durability Perspective," *ICIC Express Letters-An International Journal of Research and Surveys*, vol. 12, no. 6, pp. 615-620, 2018.
[26] R. Kumar, et al., "Durability Challenges in Software Engineering," *CrossTalk-The Journal of Defense Software Engineering*, pp. 29-31, 2016.
[27] R. Kumar, et al., "Security Assessment through Fuzzy Delphi Analytic Hierarchy Process," *ICIC Express Letters-An International Journal of Research and Surveys*, vol. 12, no. 10, pp. 1053-1060, 2018.
[28] K. Sahu and R. Shree, "Helpful and Defending Actions in Software Risk Management: A Security Viewpoint," *Integrated Journal of British*, vol. 2, pp. 1-7, 2015.
[29] A. K. Pandey, et al., "A Framework for Producing Effective and Efficient Secure Code through Malware Analysis," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 2, pp. 497-503, 2020.
[30] R. Kumar, et al., "Revisiting Software Security: Durability Perspective," *International Journal of Hybrid Information Technology*, vol. 8, no. 2, pp. 311-322, 2015.
[31] S. A. Khan, et al., "Evaluating Performance of Software Durability through an Integrated Fuzzy-Based Symmetrical Method of ANP and TOPSIS," *Symmetry*, vol. 12, no. 4, pp. 493-507, 2020.
[32] R. Kumar, et al., "Revisiting Software Security Risks," *British Journal of Mathematics & Computer Science*, vol. 11, no. 6, pp. 1-10, 2015.
[33] R. Kumar, et al., "Software Security Testing: A Pertinent Framework," *Journal of Global Research in Computer Science*, vol. 5, no. 3, pp. 23-27, 2014.
[34] R. Kumar, et al., "Software Security Durability," *International Journal of Computer Science and Technology*, vol. 5, no. 2, pp. 23-26, 2014.
[35] W. Schwittek, et al., "A common body of knowledge for engineering secure software and services," in *2012 Seventh International Conference on Availability, Reliability and Security*, pp. 499-506, 2012.
[36] R. Kumar, et al., "Durable Security in Software Development: Needs and Importance," *CSI Communication*, vol. 39, no. 7, pp. 34-36, 2015.
[37] R. Kumar, et al., "Modern Security Challenges," *International Journal of Innovations & Advancement in Computer Science*, vol. 5, no. 4, pp. 67-72, 2016.
[38] A. Agrawal, et al., "Security durability assessment through Fuzzy Analytic Hierarchy process," *PeerJ Computer Science*, pp. 1-44, 2019.

BIOGRAPHY OF AUTHOR

Dr. Mamdouh Alenezi is currently the Dean of Educational Services at Prince Sultan University. Dr. Alenezi received his MS and Ph.D. degrees from DePaul University and North Dakota State University in 2011 and 2014, respectively. Dr. Alenezi is an associate professor in software engineering with the teaching emphasis on software engineering and software security. He participates in organizing several international scientific conferences and editorial boards of the well-reputed journals. He has extensive experience in applying data mining and machine learning techniques to solve software engineering problems. He published more than 80 papers. He conducted several research areas and development of predictive models using machine learning to predict fault-prone classes, comprehend source code, and predict the appropriate developer to be assigned to a newly reported bug. His research focuses on Software Engineering, Software Security, Machine Learning, and Open Source Software Systems.