International Journal of Electrical and Computer Engineering (IJECE)
Vol. 15, No. 3, June 2025, pp. 2911~2927
ISSN: 2088-8708, DOI: 10.11591/ijece.v15i3.pp2911-2927

A comparison of approaches for modeling software security requirements using unified modeling language extensions

Syed Muhammad Junaid Hassan[1], Aamir Shahab[2], Fatima Ali Tabba[3], Muath Alrammal[4], Fadi Abu-Amara[5], Muhammad Nadeem[6]
[1]Department of Information Technology, Balochistan University of Information Technology, Engineering and Management Sciences, Balochistan, Pakistan
[2]Department of Computer Engineering, Balochistan University of Information Technology, Engineering and Management Sciences, Balochistan, Pakistan
[3]School of Computing and Information Sciences, Sohail University, Sindh, Pakistan
[4]Department of Computer and Information Science, Higher Colleges of Technology, Abu Dhabi, United Arab Emirates
[5]Division of Applied Sciences, Cybersecurity Program, Shenandoah University, Winchester, Virginia, United States of America
[6]Computer Information Science Department, Higher Colleges of Technology, Abu Dhabi, United Arab Emirates

Article Info

Article history:

Received Jul 13, 2024
Revised Mar 5, 2025
Accepted Mar 20, 2025

Keywords:

Software security requirements
Security notations
Security symbols
UML Stereotype UMLSec
Unified modeling language

ABSTRACT

The unified modeling language (UML) supports extension mechanisms called stereotypes, tagged values, and constraints to extend its modeling capabilities. These extension mechanisms are utilized to create new and customized profiles. Their applications in modeling emerging security requirements are discussed. To model authentication, availability, integrity, access control, confidentiality, data integrity, non-repudiation, authorization, encryption, hashing, and session mechanisms, a set of novel stereotypes is proposed in this paper. The proposed stereotypes inherit from baseline security requirements. Further, security concepts within the UML diagram are represented using these stereotypes and security notations. In addition, the proposed stereotypes were evaluated with the help of human subject evaluation using real-world scenarios to illustrate the usefulness of these stereotypes in modelling security requirements. This paper contributes a stereotyped model for security requirements with security symbols and a library of high-quality security notations, which can be integrated into existing or new diagrams to enhance security requirements modeling. Results indicate that the proposed stereotyped model improves the modeling process of security requirements. It also provides a better representation of emerging security mechanisms in software design. Finally, during the software development process, stakeholders enjoy improved communication and understanding of security requirements.

Corresponding Author:

Aamir Shahab
Department of Computer Engineering, Balochistan University of Information Technology, Engineering and Management Sciences
Balochistan, Pakistan
Email: aamir@ieee.org

1. INTRODUCTION

UML stereotypes have been used to model the different types of security requirements. Examples of security requirements include authenticity, secrecy, integrity, secure communication, fair exchange, non-repudiation, freshness, guarded access, secure information flow, and role-based access control. Since threats to computing systems are increasing day by day, security requirements are also changing accordingly. To meet

these security requirements, new mechanisms and technologies have been introduced [1]. Hence, there is a need to define new stereotypes and/ or other UML extensions that help software designers to design software in a more meaningful and smart way. In this paper, we discuss risk-based authentication, multi-factor authentication, two-factor authentication mechanisms, and other security requirements which are explained in section 2, and propose security symbols to model security requirements related to them.

Two-factor authentication (TFA) is a type of multi-factor authentication that requires the presentation of two or more authentication factors [2]. These factors usually include something the user knows (e.g. password or PIN), something the user has (e.g., a credit card or a smart card), and/or something the user is (e.g., a thumb impression or any other biometric feature).

Risk based authentication is a dynamic authentication mechanism that uses the agent profile that requests access to the system to determine the risk profile associated with that transaction. The risk profile is then used to find the authentication technique or intricacy of the challenge to be used. Higher-risk profiles lead to stronger challenges, whereas simple username/password authentication may be considered sufficient for lower-risk profiles.

Many software developers lack awareness and sufficient training in security. In addition, many programming books do not teach how to write secure programs [3]. Research in this domain has led to the creation of tools and prototypes, though only a limited number of them are accessible to the public. Furthermore, the suggested notations have undergone diverse evaluations, including case studies, scenarios, and user experiments, to showcase their efficacy. The rest of this paper is organized as follows. Section 2 discusses the proposed approach; section 3 discusses the literature review; section 4 discusses the results; and section 5 provides the conclusions.

2. RELATED WORK

Researchers have suggested diverse modeling notations to depict security requirements; however, these notations have not been integrated into widely used software modeling tools, and their efficacy remains unassessed. A deep learning model was developed to protect users from phishing attacks by creating a smart warning system, achieving a high accuracy of 98.4%. A multiscale approach using a Python-based scanner was presented for the detection and mitigation of web-based applications. Lack of security in virtual machines within the cloud compromises data integrity, confidentiality, and availability. Researchers proposed a MobileNet-based approach to enhance data security in the cloud with higher accuracy.

Muneer *et al.* [4] integrated security notations into a unified tool, followed by the evaluation of their effectiveness through experiments involving human subjects. Ashraf *et al.* [5] proposed guidelines for the security assessment of an enterprise applications which were analyzed in a better way. Different algorithms were introduced to secure sensitive information and personally identifiable data for cloud security. Among them, advanced encryption standard was used which encrypts a large amount of data, and is more efficient. A secure model was proposed to detect attacks on web-based applications. This CapsuleNut model includes decoding, generalization, standardization, and vectorization.

2.1. Security requirements using risk assessment techniques

Attack trees use a tree structure to model attacks where the root represents the target and leaves represent means of achieving the target. To design an attack tree, the attacker's characteristics must be considered [6]. McDermott modeled attack nets using petri nets [7]. An attack's progress is represented by means of tokens that can move along the arcs. Attack graphs are used for modeling computer network vulnerabilities, and directed graphs are used for modeling attack graphs [8]. Vulnerabilities, systems, and users are illustrated by nodes, where attacker actions are shown with edges. Use cases were introduced to model a system's unwanted behavior. This technique is referred to as a misuse case. Misuse cases are used to predict an attack. Figure 1 shows a misuse case model indicating a hacker's attack, where security can be compromised by exploiting the authentication process using a valid account or by means of a dictionary attack [9].

Use cases can be used to depict how a user can harm the system. This method was termed as an abuse case [10]. In this model, a legitimate user can perform malicious activities by using the personal privileges provided. Figure 2 shows how a malicious student can exploit a lab system. Nevertheless, system quality requirements engineering (SQUARE), is a 9-step process for elaborating an information system's security requirements [11]. The goal is to develop a view of security requirements in the initial stages of development. UMLSec, an extension to UML [12], uses stereotypes and adversary models [13], to incorporate information related to security into UML diagrams. The focus of SecureUML is on distributed system policies and control. Additionally, the approach does not propose any notations, but rather a security annotation form. The extended UML states chart notation introduces different states such as threatened,

vulnerable, defensive, compromised, quarantine, and recovery [14]. Human subjects-based evaluation is used to validate the proposed notation.

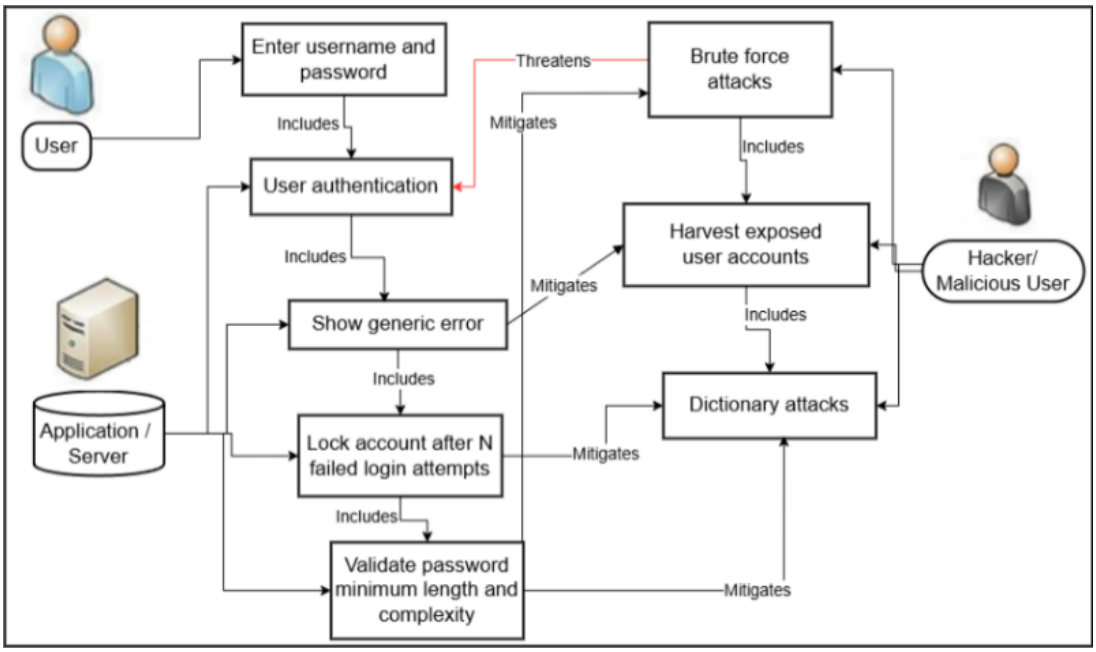


Figure 1. A misuse case model of an attack [9]

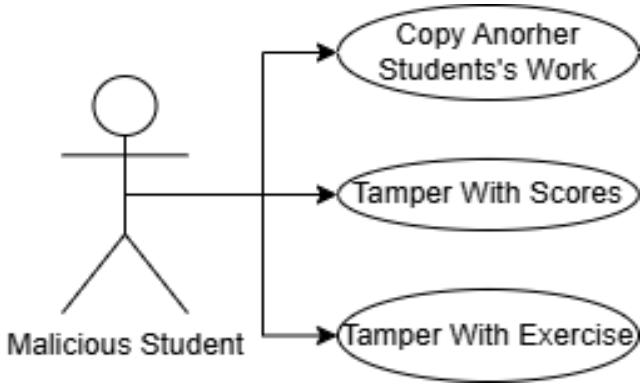
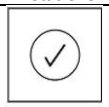


Figure 2. An abuse case model of an attack [10]

2.2. Security requirements using security notations

Existing notations of Business Process Modeling Notation (BPMN) do not represent availability, integrity, and confidentiality. Therefore, it has been extended to provide secure healthcare processes. This scheme can be used to represent events such as data scanning or running encryption algorithms. The authors have proposed seven security events for conceptual representation of authentication, authorization, harm protection, access control, non-repudiation, encrypted messages, and secure communication in the context of BPMN [15]. Different colors were used to represent security code notations. Among them, white represents normal activity diagrams, red with dotted lines represents attack activities, and blue represents defensive activity components [16]. Some notations are confusing to interpret and they can convey multiple meanings. A study was conducted to determine the usability of the notation. SecBPMN includes security notations for BPMN, such as availability, confidentiality, integrity, auditability, accountability, non-repudiation, authenticity, and privacy [17]. An experiment was conducted in which a set of security notations were selected as a result of a student-drawn set of notations for certain security concepts. Validation of these notations was done by experts. The proposed security notations can be seen in Table 1.

Table 1. Security concepts and notations in business processes [18]

Security concepts	Notations	Security concepts	Notations	Security concepts	Notations
Availability		Access control		Privacy	
Data confidentiality		Data integrity		Risk	
Audit		Encryption		Digital signature	

This study is limited to business processes, and a study is needed that explains the results in terms of the total number of notation, security concepts, and languages used for modeling. Moreover, tool support is missing and the security concepts chosen were also limited [18]. Rodríguez *et al.* [15] presented an extension to BPMN without providing any supporting tool. However, they took an example of a healthcare scenario to prove the concept. In the study, no security experts were involved nor additional studies were carried out. In [19], business processing notation is presented by simply extending BPMN. In addition, when designing these notations, the safety and security of assets in the food industry was considered. The author divided assets into two branches: physical and logical as shown in Table 2.

A limited set of notations is selected that is applicable for security purposes. Maines *et al.* argue that notations used for security must be a true representation so that they can be easily implemented [20], *e.g.*, using a wall with fire to represent a firewall. This displays the true essence of the concept as shown in Table 3. Organizational trust and security groups were added to BPMN [21]. These notations are an abstract level representation of state security requirements. However, no studies have been conducted, and the set of proposed notations is also limited. Similarly, a limited number of notations were presented in [22], [23]. The proposed notations for workflow systems are shown in Table 4. Security notations such as integrity, availability, and confidentiality were presented by Altuhhov [24].

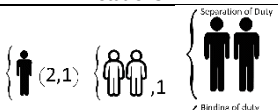
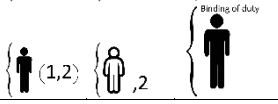
Table 2. Control for physical and logical assets [19]

Physical assets	Notation	Logical assets	Notation
Deep-Frozen		Private	
Light-Sensitive		Financial	
Explosive		Legal	
Poisonous		Confidential	
Radioactive		Audit-Relevant	

Table 3. Cyber-security ontology for BPMN [20]

Security concept	Notation
Firewall	

Table 4. Security notations for work-flow [23]

Security concept	Notations
Separation of duty	
Binding of duty	

The author's argument regarding notations is that they lack attention to security concerns. This research focuses on complete mediation, policies, cryptography, credentials, and interception. A tool was developed to gather various conceptual patterns from literature. While many notations were described based on texts, their visual representation was missing [25]. The MASC modeling architecture can be viewed in Table 5.

Security is not a separate concern that should be considered only in the initial stages of the requirement gathering process. The author proposed a set of notations that can be implemented using UML and other modeling tools [26]. These notations can be used for designing the security of client-server systems [27]. Business designers elicited the security requirements at a conceptual level, which were later implemented by developers based on those requirements. Similarly, a UML-based tactic for SOA applications was presented in [28], as shown in Table 6.

An activity diagram was used to incorporate proposed security notations, and a proof of concept was given by depicting a healthcare scenario [29]. As discussed earlier, similar notations were proposed for UML, like BPMN. The proposed notations are displayed in Table 7.

Table 5. MASC modeling architectural security concepts [25]

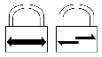
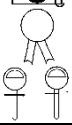
Security concepts	Notations	Security concepts	Notations	Security concepts	Notations
Session		Data is stored		Enforcement	
Session initialized		Session closed		Administration	
Encryption		Signing		Decision	
Hashing		Keys		Information	

Table 6. SOA security notations [28]

Security concepts	Notations	Security concepts	Notations	Security concepts	Notations
Data integrity		Non-repudiation		Authorization	
Availability		Authentication		Traceability and auditing	

Table 7. UML security modeling extensions [29]

Security concepts	Notations	Security concepts	Notations	Security concepts	Notations
Security auditing		Security requirement		Non-repudiation	
Integrity		Access control		Privacy	

2.2.1. Independent modeling notations

In [30], a visual language for authorization controls was discussed. The main notation used to represent authorization is a piece of paper with a signature, and a few supporting notations were also proposed. Table 8 shows the notations of the visual language for authorization modeling [31]. The notations are vague when used independently, as their meaning is unclear and no tools were provided. Furthermore, no user studies were conducted. In contrast, extensive user studies were conducted using a larger set of notations. Literature indicates that de facto software modeling tools have not incorporated these security notations. This implies the need to incorporate these security notations in a unified way which is the objective of this study.

Table 8. Visual language for authorization modeling [30]

Security concepts	Notations	Security concepts	Notations	Security concepts	Notations
Authorization on a form		Valid		Used	
Valid and put on hold		Invalid		Unused	

2.3. UML extension mechanism

The UML extension mechanism allows developers to customize and enhance standard UML models to better fit specific domains or requirements. It includes constructs like stereotypes, tagged values, and constraints, which enable the addition of new semantics or properties to existing UML elements without altering the core UML specification. This flexibility supports domain-specific modeling, such as incorporating security, performance, or platform-specific concerns directly into diagrams. In this section, we will discuss these mechanisms with examples.

2.3.1. Constraints

They are used to create new semantics, properties for specifying semantics, and/or conditions. They are specified by using a string enclosed by brackets. Figure 3(a), shows a simple example of using constraints in a UML model. This UML class diagram represents a system with individuals, financial accounts, and organizations, incorporating rules to ensure data integrity and security. The Person class includes gender and optional spouse associations, with constraints enforcing heterosexual relationships. The BankAccount class is linked to both a portfolio and a required corporation, and is marked with a {secure} tag to highlight the need for security measures. Overall, the diagram outlines key structural relationships and embeds business rules alongside security considerations.

2.3.2. Tagged values

They allow the creation of new properties specified in the form of keyword-value pairs and extend UML building blocks by creating new information for the element. Tagged values can be created for existing model elements or stereotypes. Tagged values are not class attributes. Figure 3(b), demonstrates the use of tagged values. ”Version=3.2” is a tagged value and not an attribute of the class. The tagged value in this example may be useful for software developers.

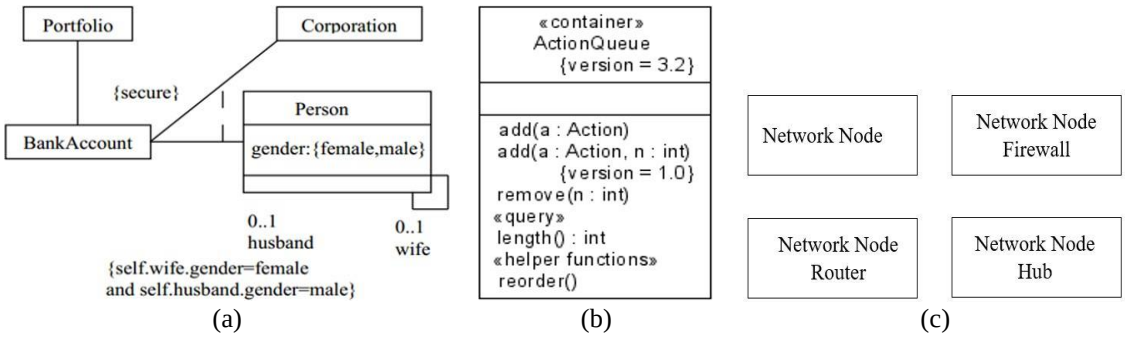


Figure 3. UML extension mechanism constructs: (a) constraints, (b) tagged values, and (c) stereotypes

2.3.3. Stereotypes

Allow the extension of the vocabulary of UML to create new model elements derived from existing ones. The newly created elements have specific properties suitable for a particular problem domain. Graphically, a stereotype is rendered as a name enclosed by guillemets. Figure 3(c), shows some examples of stereotyped modeling elements.

By using the ‘network node’ stereotype, the firewall, router, and hub may be used as first-class citizens when modeling the system, thus adding more semantics to the design. With the impact of rapid technological changes, cybersecurity-related concepts are difficult to understand. Vallabhaneni *et al.* proposed a bidirectional generative adversarial network approach for the detection of cyber-attacks [32]. Visual notations can be used for cyber security research. Sturdee *et al.* makes a collection of sketches and

icons which function as the foundation for creating a visual language for interfaces and communication related to cybersecurity [33].

As UMLsec does not explicitly addresses access control, an approach was proposed to resolve the mentioned challenges by enhancing UML through the inclusion of security diagrams, which will serve to depict mandatory access control (MAC), discretionary access control (DAC), and role based access control (RBAC) policies as integral components [34]. Basin *et al.* proposed an approach called model driven security to build a secure system. Instead of prescribing a specific modeling language for this procedure, they suggested a comprehensive framework for developing such languages by integrating systems modeling languages with security modeling languages [35]. They showcase various examples of this framework, demonstrating the synthesis of various UML modeling languages with a security modeling language to articulate access control requirements in a formal manner.

2.4. Security requirements and UMLSec

This section briefly discusses the security requirements that are needed to be achieved while modeling any secure system. The following security requirements have been identified by [36], and are mandatory to achieve for any system to be secure from threats. Key security requirements supported by UMLSec include fair exchange, secure links, secure dependencies, guarded access, no down-flow and no up-flow policies, ensuring that both structural and behavioral aspects of security are considered during system development. Security Requirements for threat free systems are mentioned in Table 9. The UML stereotypes used for these security requirements will be discussed in the next section.

Table 9. Description of security requirements

Security requirement	Description
Fair exchange	Prevent the participants from cheating
Non-repudiation	The action cannot subsequently be denied
Role based access control	A mechanism for controlling access to protected resources, assign permissions to roles, instead of assigning to people
Secure communication link	Preserve ‘Secrecy’ and ‘Integrity’ of the data in transit
Secrecy and integrity	Data should be read only by the legitimate parties (Secrecy), Data should be modified only by the legitimate parties (Integrity)
Authenticity	Message authenticity, data origin authenticity, entity authenticity
Freshness	Message created in the current execution round must not be a replay of an older message by the adversary
Secure information flow	A concept used in multi-level secure systems having different levels of sensitivity of data (e.g., high and low). High means highly sensitive or highly trusted data. What happens when highly trusted parts communicate with low trusted parts and vice versa? A system may have ‘No down-flow’ and/or ‘no up-flow’ policies
Guarded access	Access control, ensure that only legitimate parties have access to secure part of the system

3. PROPOSED APPROACH

This section discusses the proposed approach. The proposed approach comprises of 4 steps, the first step deals with extraction of symbols and notations, second step deals with selection of appropriate notation for representation of a security concept, step three deals with conversion of these notations and security concepts to stereotypes and finally the step four deals with Evaluation of stereotypes through depiction of a scenario. This proposed 4 steps approach is depicted in Figure 4 (a), (b), (c), and (d). As a first step, security symbols and notations used to represent various basic security concepts were collected from literature. A set of 94 security notations and symbols were gathered, some of which are discussed in the related work section due to limitations of space; all of them cannot be discussed here but are available in the online repository. The notations were then redrawn in high quality as most of them were not available publicly and in usable format. After accounting for similarity and considering basic security concepts out of 94 still 49 security notations and symbols were remaining.

In the second step, security symbols and notations were divided into groups according to the security concepts they were representing and were presented to the participants of survey which was conducted on Amazon mechanical Turk. The participants were selected if specific conditions were met, the participant must be working in the software industry for more than 5 years. Secondly, participants must also have knowledge of different types of security concepts and their applications. The participants must also have knowledge of different types of modelling languages such as UML, BPMN and others. The survey was conducted to select the most appropriate visual representation of a security concept because different researchers have presented the same security concept differently. A sample question from the questionnaire is shown in Figure 5, where the respondent is asked to select the most appropriate symbols for the concept of session.

In the third step, the final set of notations and symbols were used to generate UML stereotypes. The selected notations satisfy most of the security requirements of secure system which are discussed in the previous section. Table 10 shows the mapping of how selected notations are related to security concepts, baseline requirements, security requirements and how they can be transformed into stereotypes and where these UML stereotypes can be applied.

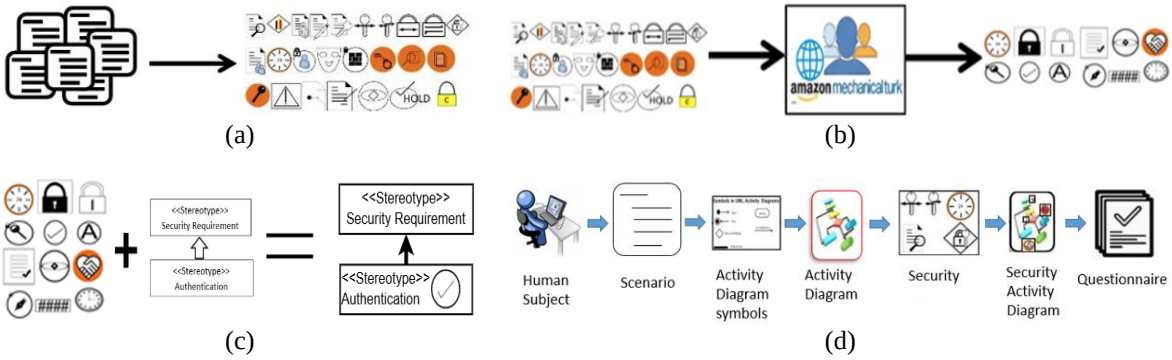


Figure 4. Steps of proposed approach (a) step 1 symbol/notation extraction, (b) step 2 best visual representation of security concept, (c) step 3 security concept to stereotypes, and (d) step 4 evaluation of security stereotypes

1. Which of the following symbol best describe "Session"?

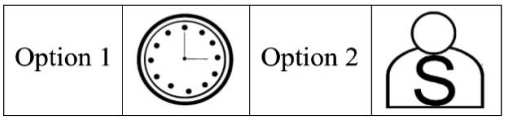


Figure 5. Sample question

Table 10. Mapping of security concepts, security requirements, and UML stereotypes

Security concept	Baseline requirement (s)	Stereotype	Applies to	Related security requirement
Availability	Availability	<<*Availability*>>	Components, nodes	-
Integrity	Integrity	<<*Integrity*>>	Data, classes	Secrecy and integrity
Access control	Confidentiality, integrity, authorization	<<*AccessControl*>>	Interfaces, services	Role based access control, guarded access
Confidentiality	Confidentiality	<<*Confidentiality*>>	Data, classes	Secrecy and integrity
Data integrity	Integrity	<<*DataIntegrity*>>	Data, classes, attributes, associations	Secrecy and integrity
Non-repudiation	Accountability (integrity)	<<*NonRepudiation*>>	Logs, operations	Non-repudiation
Authentication	Identification & authentication	<<*Authentication*>>	Services, interfaces	Authenticity
Authorization	Access control (confidentiality)	<<*Authorization*>>	Methods, services	Role based access control, guarded access
Encryption	Confidentiality, integrity	<<*Encryption*>>	Communication links	Secure communication link
Hashing	Integrity	<<*Hashing*>>	Attributes, data	Integrity
Session	Availability, confidentiality	<<*Session*>>	Components, interfaces	-
Sensory experience	Authentication (usability & biometrics)	<<*SensoryExperience*>>	Actors, devices	-

UML stereotypes proposed in [29] were used as the base class and new classes that inherit from the stereotype 'security requirement' were proposed. Figure 6 shows one of the stereotypes proposed in our approach. Due to the limitation of space only stereotypes related to authentication are displayed here. The stereotype 'authentication' is defined as a child of 'security requirement'. The stereotype 'authentication' has

two children, namely ‘multi-factor authentication’ and ‘risk- based authentication’ stereotypes. The first one has yet another inherited stereotype named ‘two-factor authentication’.

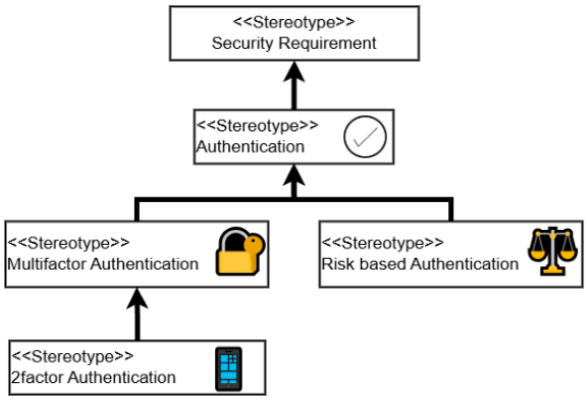


Figure 6. The proposed stereotypes

3.2. Sample model for risk based authentication

Assume that a client accesses an online banking system using different types of networks and devices. The network may be a public wireless network, a private secure network, or a mobile network. The client may use a recognized device (e.g., the user’s laptop), an unregistered public computer at an airport, or a mobile device with internet access. The online banking system uses the risk profile associated with the type of access and determines the level of challenge to be used while authenticating the user. If the user tries to access their online banking account from an unrecognized device, they may be asked to answer more questions than just their user ID and password. This scenario has been depicted in Figure 7.

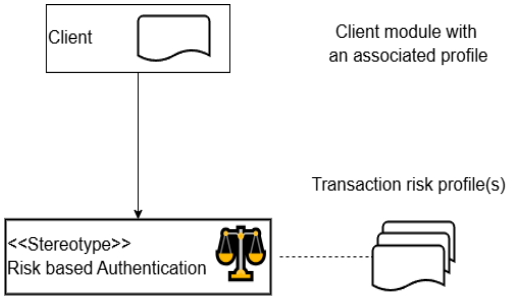


Figure 7. Risk based authentication

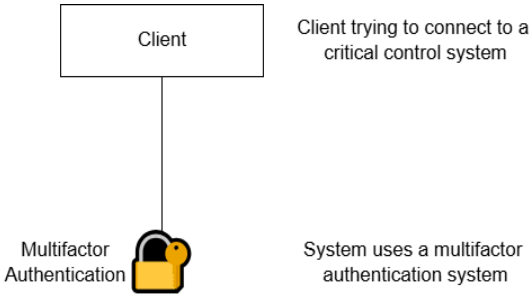


Figure 8. Multi factor authentication

3.3. Sample model for multi-factor authentication

Multi-factor authentication uses more than two factors for authentication. Usually, these factors include what you know (e.g., username/password or PIN), what you have (e.g., an ATM card, OTP device or a smart card), and finally, what you are (e.g., fingerprint, retina scan or facial recognition). By combining multiple forms of authentication, multi-factor authentication significantly enhances security, making it much harder for unauthorized individuals to access sensitive information, even if they have obtained a password. It is widely used to protect online accounts, corporate systems, and sensitive data from cyber threats.

To demonstrate the use of the proposed multi-factor authentication system, let’s assume a user of a critical control system. When the user wants to log on to the system, they use their smart card as the first step of the authentication process, and later they are asked to enter their password. The scenario has been depicted in Figure 8. The client module provides smart card information to the authentication module. Upon success, the multi-factor authentication module further requests the client to send their username and password to complete the authentication process.

Similarly, the stereotypes for other security concepts were created. Finally, in the fourth step, the UML stereotypes were tested with the help of human subject evaluation where human subjects were given a

scenario to represent graphically. The scenario presented to human subjects was curated by security experts. Certain security concepts were included in the scenario so that UML stereotypes can be tested. Either stereotype name or symbol or both can be used to represent the security requirement. In the first attempt the respondents were not exposed to UML stereotypes with security symbols, after the first attempt they were provided with the UML stereotypes with security symbols. At the end, respondents were requested to fill a questionnaire regarding usability and usefulness of proposed UML stereotypes.

4. RESULTS AND DISCUSSION

To analyze security symbols to be used in proposed stereotypes a survey was conducted on Amazon mechanical Turk. The purpose of this survey was to reduce the number of symbols available to be used in UML stereotypes as many researchers used different symbols/notations to represent the same security concept. This survey resulted in a final set of 12 security symbols/notations visible in Figure 9, which were then converted in stereotypes and incorporated in Microsoft Visio stencil and Draw.io library so that they may be used in the fourth step of the study.

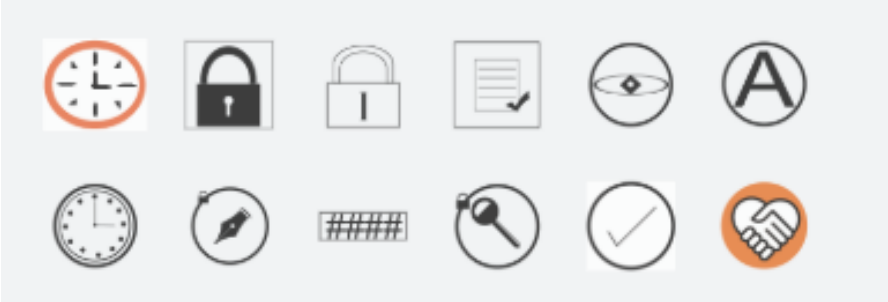


Figure 9. Final set of security symbols and notations

In the fourth step, the library and stencil of UML stereotypes combined with security notations and symbols were shared with the human subjects. They were asked to draw diagrams with security symbols and notations and without security symbols and notations. The primary objective was to evaluate how well different security notations were understood and implemented by software developers and designers. Responses were conventional diagrams that incorporate security symbols. They provide valuable insights into how the security requirements can be modelled by extending UML through stereotypes. Moreover, the survey conducted at the end also presented valuable results in terms of usability of these proposed stereotypes.

Figures 10 to 15 are displaying the typical scenario encountered while trying to access a bank account and perform a transaction the description of which is as follows. The bank wants to facilitate its customers to perform financial transactions 24/7 ubiquitously. When The customer opens the bank's website, the customer will be provided with safety information about the account and how to use it in a pop up window. The customer is then required to enter a user name and to provide proof that the customer is human and not a machine by entering characters and figures/numbers that appear in the form of an image. After successful verification, the customer is granted access to the next page where password is required. According to the bank's security policy customer are never asked to enter a complete password but only a part of it. After successful verification, login process is complete as shown in Figure 10 and Figure 11, accessing bank account with/without security symbol respectively.

After login, the customer is granted access to a bank account where different transactions can be performed. If the customer remains active for more than 5 minutes, the customer will be automatically logged off from the bank's website. When the customer initiates a financial transaction, say for example a transfer of funds to someone's account, a verification is again required that it is the customer who has initiated the transaction by providing a one-time authorization code which is sent either to customer's email address or mobile number. After successful verification, the transaction is completed and the customer is notified. Customers can perform other transactions also, for example paying utility bills. Different responses with security symbols are shown in Figure 13 and 15 and without security symbols for accessing bank account are shown in Figure 12 and 14.

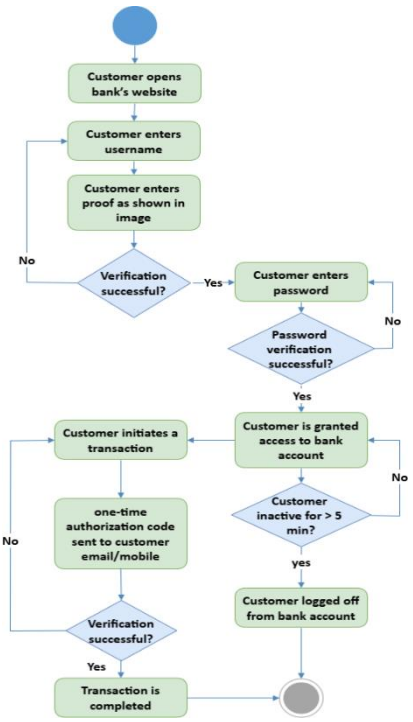


Figure 10. Response 1a: accessing bank account without security symbols

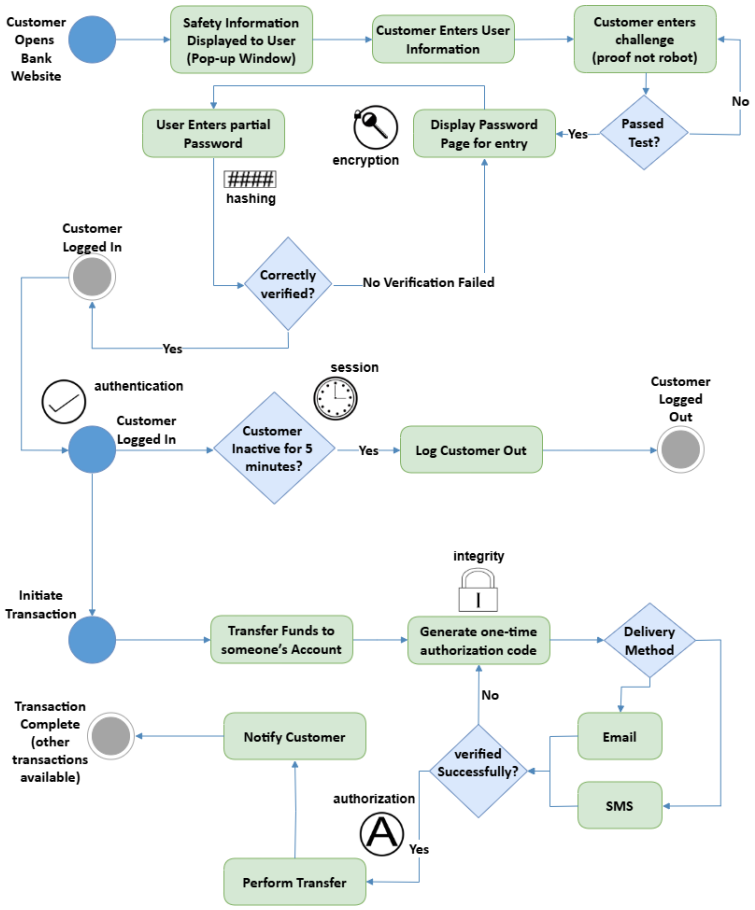


Figure 11. Response 1b: accessing bank account with security symbols

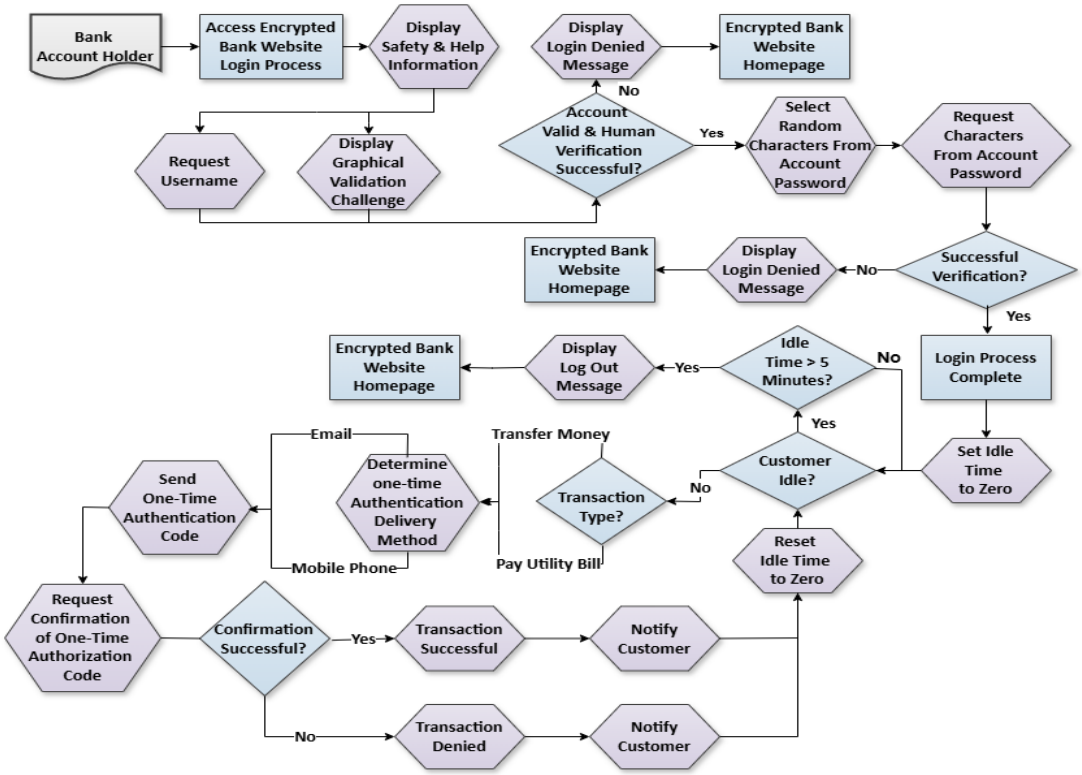


Figure 12. Response 2a: accessing bank account without security symbols

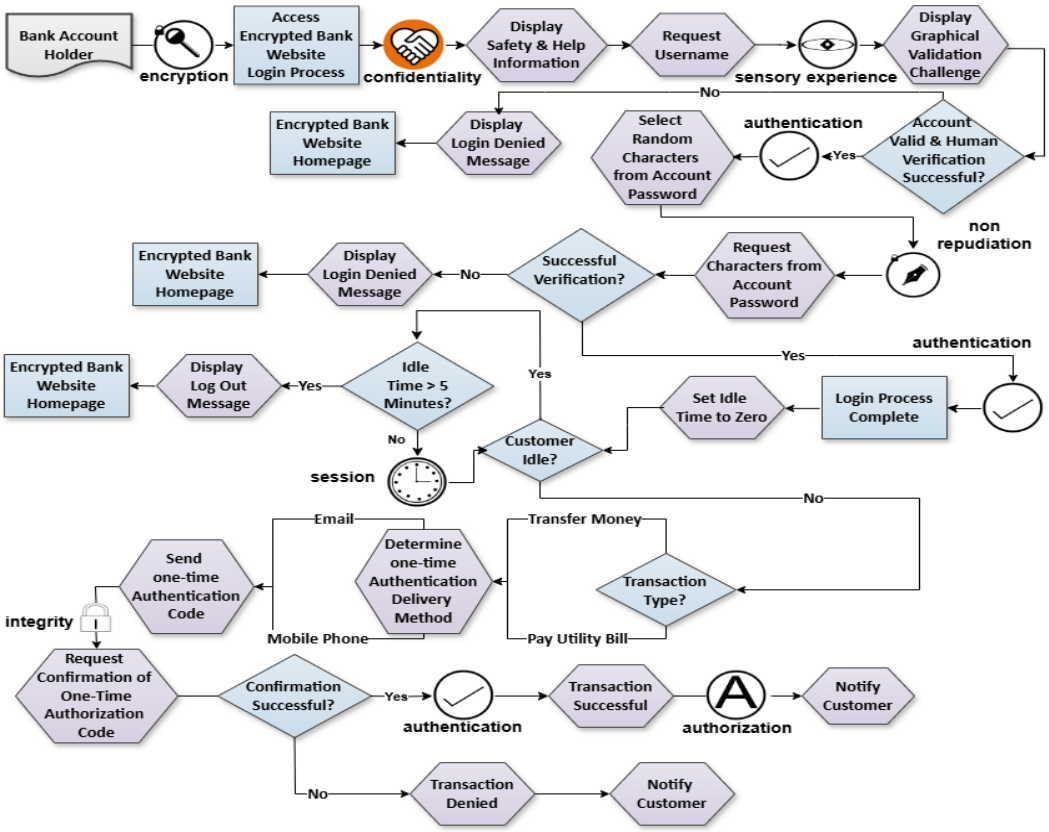


Figure 13. Response 2b: accessing bank account with security symbols

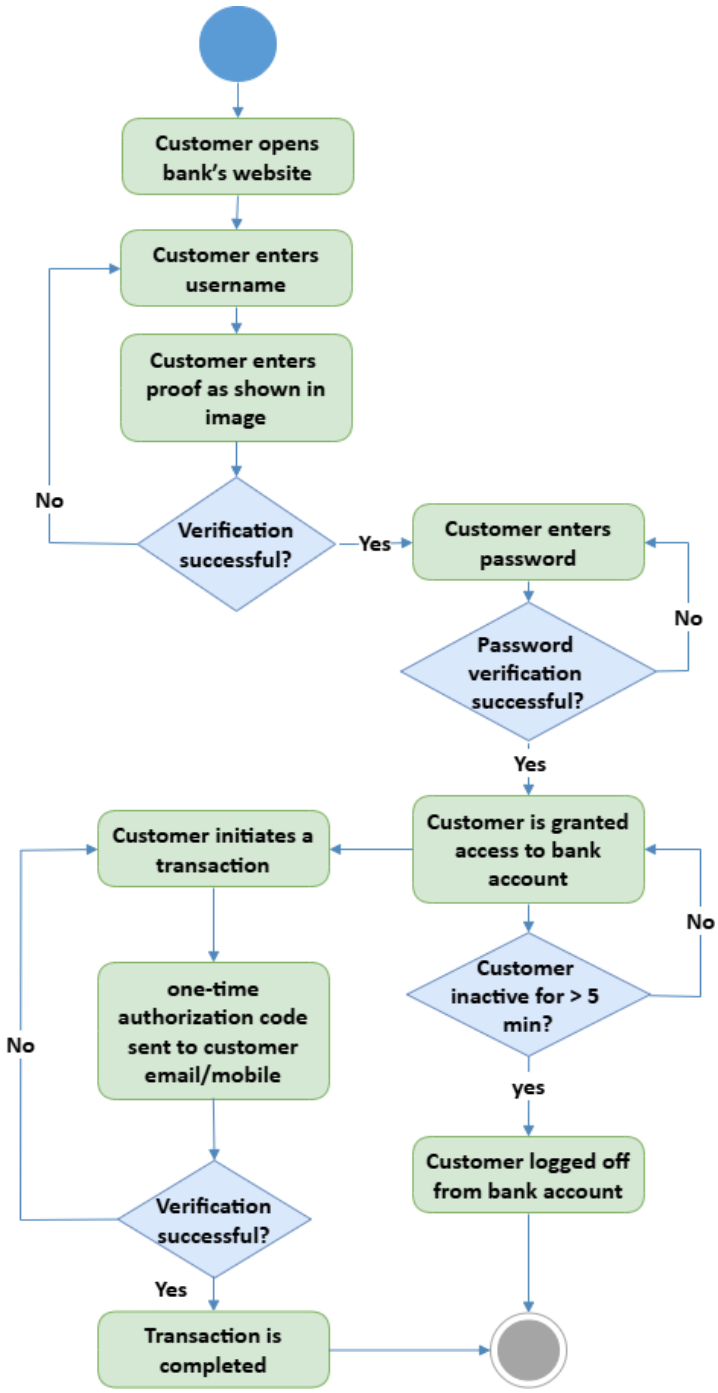


Figure 14. Response 3a: accessing bank account without security symbols

The survey conducted at the end of activity provided favorable results. 90% respondents supported the usefulness of UML stereotypes with security notations. 80% respondents stated that there should be more UML stereotypes with security notations. 75% respondents suggested that the provided UML stereotypes with security notations are easy to learn and use. 80% respondents stated that UML stereotypes with security notations helped in depicting the provided scenario. Overall, it can be inferred that the extension of UML stereotypes to represent security concepts through security notations is a valuable tool for modelling security requirements.

As a result, unique artifacts were collected as the fourth and final part of the study concluded. Artifacts with UML stereotypes having security notation and symbols are more descriptive and easier to perceive than artifacts without security notation and symbols. Results indicate that the proposed stereotyped model improves the modeling process of authentication related security requirements and other security

requirements and concepts. It also provides a better representation of emerging security mechanisms in software design. Finally, during the software development process, stakeholders enjoy improved communication and understanding of security requirements.

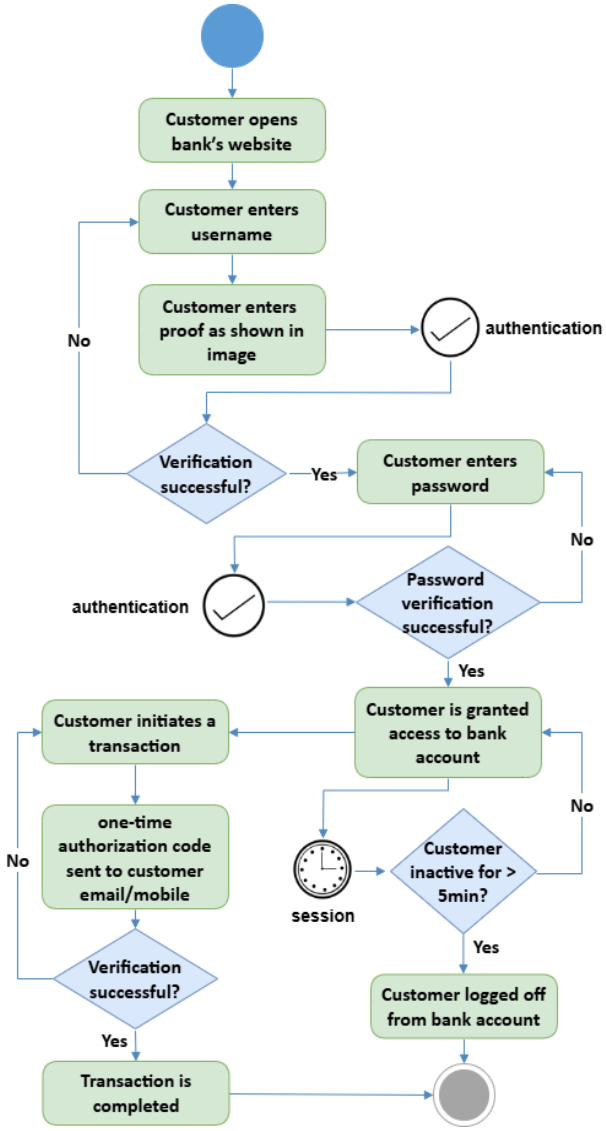


Figure 15. Response 3b: accessing bank account with security symbols

5. CONCLUSION

In this paper, we discuss the extensibility features of the Unified Modeling Language and how different security requirements are fulfilled by using UML stereotypes. Based on the work already done, we propose new stereotypes with security symbols and notations, namely authentication, availability, integrity, access control, confidentiality, data integrity, non-repudiation, authorization, encryption, hashing, and session. Furthermore, the UML stereotypes with security symbols is a great tool for software engineers for representing security requirements. However, the security symbol library that is made available as one of the deliverables is a valuable contribution of this study and can be utilized to produce UML stereotypes and diagrams that support other security requirements mentioned not covered in this study.

FUNDING INFORMATION

No funding was involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Syed Muhammad Junaid Hassan	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Aamir Shahab		✓	✓		✓	✓		✓		✓	✓	✓		
Fatima Ali Tabba	✓					✓	✓							
Muath Alrammal						✓	✓							✓
Fadi Abu-Amara						✓	✓						✓	✓
Muhammad Nadeem	✓			✓	✓	✓	✓			✓	✓	✓		✓

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

No conflict of interest is reported among authors.

INFORMED CONSENT

Informed consent was obtained from all individuals included in this study.

DATA AVAILABILITY

The data that support the findings of this study are openly available in GitHub repository named "A-comparison-of-approaches-for-modeling-software-security-requirements-using-UML-extensions-IJECE" at https://github.com/smjunaid-it/A-comparison-of-approaches-for-modeling-software-security-requirements-using-UML-extensions-IJECE].

REFERENCES

[1] O. Pilskalns, D. Williams, D. Aračić, and A. Andrews, "Security consistency in UML designs," in *Proceedings - International Computer Software and Applications Conference*, 2006, vol. 1, pp. 351–358. doi: 10.1109/COMPSAC.2006.76.

[2] A. H. Y. Mohammed, R. A. Dziyauddin, and L. A. Latiff, "Current multi-factor of authentication: Approaches, requirements, attacks and challenges," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 1, pp. 166–178, 2023, doi: 10.14569/IJACSA.2023.0140119.

[3] A. Shahab, M. Nadeem, M. Alenezi, and R. Asif, "An automated approach to fix buffer overflows," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 4, pp. 3777–3787, Aug. 2020, doi: 10.11591/ijece.v10i4.pp3777-3787.

[4] S. U. Muneer, M. Nadeem, B. Kasi, and M. H. Yousaf, "Evaluating the effectiveness of notations for designing security aspects," *International Conference on Advanced Communication Technology, ICACT*, vol. 2020, pp. 465–471, 2020, doi: 10.23919/ICACT48636.2020.9061305.

[5] H. Ashraf, M. Alenezi, M. Nadeem, and Y. Javid, "Security assessment framework for educational ERP systems," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 6, pp. 5570–5585, 2019, doi: 10.11591/ijece.v9i6.pp5570-5585.

[6] B. Schneier, "Attack trees," *Dr. Dobb's journal*, vol. 24, no. 12, pp. 21–29, 1999.

[7] J. P. McDermott, "Attack net penetration testing," in *Proceedings New Security Paradigm Workshop*, 2000, pp. 15–21. doi: 10.1145/366173.366183.

[8] C. Phillips and L. P. Swiler, "A graph-based system for network-vulnerability analysis," *Proceedings New Security Paradigms Workshop*, vol. Part F1292, pp. 71–79, 1998, doi: 10.1145/310889.310919.

[9] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Engineering*, vol. 10, no. 1, pp. 34–44, 2005, doi: 10.1007/s00766-004-0194-4.

[10] J. McDermott and C. Fox, "Using abuse case models for security requirements analysis," in *Proceedings - Annual Computer Security Applications Conference, ACSAC*, 1999, vol. Part F1334, pp. 55–64. doi: 10.1109/CSAC.1999.816013.

[11] N. R. Mead and T. Stehney, "Security quality requirements engineering (SQUARE) methodology," in *SESS 2005 - Proceedings of the 2005 Workshop on Software Engineering for Secure Systems - Building Trustworthy Applications*, 2005, pp. 1–7. doi: 10.1145/1083200.1083214.

[12] T. Lodderstedt, D. Basin, and J. Doser, "SecureUML: a UML-based modeling language for model-driven security," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2460

LNCS, pp. 426–441, 2002, doi: 10.1007/3-540-45800-x_33.
[13] J. Jürjens, "UMLSec: Extending UML for secure systems development," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2460 LNCS, pp. 412–425, 2002, doi: 10.1007/3-540-45800-x_32.
[14] M. El-Attar, H. Luqman, P. Karpati, G. Sindre, and A. L. Opdahl, "Extending the UML statecharts notation to model security aspects," *IEEE Transactions on Software Engineering*, vol. 41, no. 7, pp. 661–690, 2015, doi: 10.1109/TSE.2015.2396526.
[15] A. Rodríguez, E. Fernández-Medina, and M. Piattini, "A BPMN extension for the modeling of security requirements in business processes," *IEICE Transactions on Information and Systems*, vol. E90-D, no. 4, pp. 745–752, 2007, doi: 10.1093/ietisy/e90-d.4.745.
[16] M. N. Gedam and B. B. Meshram, "Proposed secure activity diagram for software development," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 6, pp. 671–680, 2023, doi: 10.14569/IJACSA.2023.0140671.
[17] F. Karlsson, F. Linander, and F. von Schéele, "Modeling and verifying security policies in business processes," *Lecture Notes in Business Information Processing*, vol. 175 LNBIP, pp. 454–463, 2014.
[18] M. Leitner, S. Schefer-Wenzl, S. Rinderle-Ma, and M. Strembeck, "An experimental study on the design and modeling of security concepts in business processes," *Lecture Notes in Business Information Processing*, vol. 165 LNBIP, pp. 236–250, 2013, doi: 10.1007/978-3-642-41641-5_17.
[19] G. Monakova, A. D. Brucker, and A. Schaad, "Security and safety of assets in business processes," in *Proceedings of the ACM Symposium on Applied Computing*, 2012, pp. 1667–1673. doi: 10.1145/2245276.2232045.
[20] C. L. Maines, D. Llewellyn-Jones, S. Tang, and B. Zhou, "A cyber security ontology for BPMN-security extensions," in *Proceedings - 15th IEEE International Conference on Computer and Information Technology, CIT 2015, 14th IEEE International Conference on Ubiquitous Computing and Communications, IUCC 2015, 13th IEEE International Conference on Dependable, Autonomic and Se*, 2015, pp. 1756–1763. doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.265.
[21] M. Menzel, I. Thomas, and C. Meinel, "Security requirements specification in service-oriented business process management," in *Proceedings - International Conference on Availability, Reliability and Security, ARES 2009*, 2009, pp. 41–48. doi: 10.1109/ARES.2009.90.
[22] C. Wolter, M. Menzel, and C. Meinel, "Modelling security goals in business processes," *Modellierung 2008*, pp. 197–212, 2008.
[23] C. Wolter and A. Schaad, "Modeling of task-based authorization constraints in BPMN," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4714 LNCS, pp. 64–79, 2007, doi: 10.1007/978-3-540-75183-0_5.
[24] O. Altuhhov, R. Matulevičius, and N. Ahmed, "An extension of business process model and notation for security risk management," *Standards and Standardization: Concepts, Methodologies, Tools, and Applications*, pp. 897–919, 2015, doi: 10.4018/978-1-4666-8111-8.ch042.
[25] L. Sion, K. Yskout, A. Van Den Berghe, R. Scandariato, and W. Joosen, "MASC: Modelling architectural security concerns," in *Proceedings - 7th International Workshop on Modeling in Software Engineering, MiSE 2015*, 2015, pp. 36–41. doi: 10.1109/MiSE.2015.14.
[26] A. Bilbas, "An automated approach to classify class role stereotypes for detecting design flaws," Eindhoven University of Technology, 2024.
[27] M. Nadeem and A. H. S. Bukhari, "Notations for facilitating software security design," *Journal of Applied and Emerging Sciences*, vol. 2, no. 1, pp. 12–18, 2011, doi: 10.36785/jaes.2147.
[28] M. Q. Saleem, J. Jaafar, and M. F. Hassan, "Secure business process modelling of SOA applications using 'UML-SOA-Sec,'" *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 4, pp. 2729–2746, 2012.
[29] A. Rodríguez, E. Fernández-Medina, and M. Piattini, "Security requirement with a UML 2.0 profile," in *Proceedings - First International Conference on Availability, Reliability and Security, ARES 2006*, 2006, vol. 2006, pp. 670–677. doi: 10.1109/ARES.2006.125.
[30] S. K. Chang, G. Polese, R. Thomas, and S. Das, "A visual language for authorization modeling," in *Proceedings. 1997 IEEE Symposium on Visual Languages (Cat. No.97TB100180)*, 1997, pp. 110–118. doi: 10.1109/VL.1997.626565.
[31] S. Fluchs, R. Drath, and A. Fay, "Evaluation of visual notations as a basis for ICS security design decisions," *IEEE Access*, vol. 11, pp. 9967–9994, 2023, doi: 10.1109/ACCESS.2023.3238326.
[32] J. Jürjens, "Automated security hardening for evolving UML models," in *Proceedings of the 33rd International Conference on Software Engineering*, May 2011, vol. 35, no. 3, pp. 986–988. doi: 10.1145/1985793.1985968.
[33] M. Sturdee, L. Thornton, B. Wimalasiri, and S. Patil, "A visual exploration of cybersecurity concepts," *ACM International Conference Proceeding Series*, 2021, doi: 10.1145/3450741.3465252.
[34] S. Demurjian, J. Pavlich-Mariscal, and L. Michel, "Enhancing UML to model custom security aspects," *Proceedings of the 11th International Workshop. on Aspect-Oriented Modeling*, 2007.
[35] D. Basin, J. Doser, and T. Lodderstedt, "Model driven security: From UML models to access control infrastructures," *ACM Transactions on Software Engineering and Methodology*, vol. 15, no. 1, pp. 39–91, 2006, doi: 10.1145/1125808.1125810.
[36] J. Jürjens, *Secure systems development with UML*. Springer Science & Business Media, 2005. doi: 10.1007/b137706.

BIOGRAPHIES OF AUTHORS

Syed Muhammad Junaid Hassan received his undergraduate degree from University of Balochsitan, Quetta. He holds two graduate degrees, one in information technology and the other in business administration from the Balochistan University of Information Technology, Engineering, and Management Sciences (BUITEMS). He was awarded gold medal at undergraduate and graduate level for his academic achievements. He is serving as an assistant professor at the Department of Information Technology, BUITEMS. Currently he is pursuing Ph.D. in computing from Universiti Teknologi Malaysia, Johor Bahru, Malaysia. His research interests are software engineering, software security, software defect prediction, and application of machine learning in software engineering. He can be contacted at email: smjunaid.it@gmail.com.

Aamir Shahab holds master degree in computer engineering from BUITEMS, Quetta (2019). He obtained bachelor degree in computer engineering from Bahauddin Zakariya University, Multan (2016). He is currently serving as a lecturer computer science in BUITEMS, Quetta and pursing for his Ph.D. program in computer engineering with emphasis in cyber security. His researches are in field of software security. He is affiliated with IEEE as student member. Besides, he is interested in cyber security research and trainings. His researches are in field of software security. He is affiliated with IEEE as student member. Besides, he is interested in cyber security research and trainings. He can be contacted at email: aamir@ieee.org.

Fatima Ali Tabba is a lecturer in Sohaul University Karachi. She has been teaching in universities for more than 7 years. She was a database administrator in the IDO education program by UNICEF. She has been affiliated with the ORIC department as a research manager and conducted many workshops. She can be contacted at email: fatimaalitabba@outlook.com.

Muath Alrammal completed his M.Sc. in information technology at Telecom SudParis, Evry, France, in 2007. Following this, he attained his Ph.D. in computer sciences from the University of Paris Est, Paris, in 2011. Subsequently, he embarked on his post-doctoral journey at LACL, University of Paris Est. In 2012, he continued his research as a post-doctoral researcher at LIFO, University of Orleans, France. From 2013 to 2017, Dr. Alrammal served as an assistant professor in the IT department at KIC, Abu Dhabi, UAE. Since 2017, he has been contributing to the academic community as an assistant professor in the CIS department at HCT, Abu Dhabi, UAE. His research interests encompass a wide array of cutting-edge topics, including processing big data in streaming, performance models, selectivity estimation techniques, and machine learning. Dr. Alrammal is an esteemed member of the LaMHA research group. He can be contacted at email: malrammal@hct.ac.ae.

Fadi Abu-Amara serves as an assistant professor in the cybersecurity program, Division of Applied Technology, at Shenandoah University, USA. Dr. Abu-Amara holds a Ph.D. in electrical and computer engineering, earned in 2010 from Western Michigan University, USA. Dr. Abu-Amara brings over 22 years of experience in teaching, management, research, and industry. His research interests span diverse areas, including cryptography, gamified approaches to cybersecurity awareness, blockchain applications in energy management (electricity water), feature extraction, biomedical imaging, and leveraging robotics and virtual reality to enhance academic skills for children with autism. He can be contacted at email: fadi.abuamara@su.edu.

Muhammad Nadeem is a Fulbright alumnus and holds a Ph.D. degree in computer science from Mississippi State University, USA. He has been teaching in public sector universities of Pakistan and United Arab Emirates (UAE) for more than 15 years and has been involved in various training and consultancy projects related to software security. He was also honored with the Best University Teacher award from the Higher Education Commission of Pakistan. He can be contacted at email: dr.nadeem@ieee.org.