Hindawi
Security and Communication Networks
Volume 2022, Article ID 3083909, 22 pages
https://doi.org/10.1155/2022/3083909

Research Article

An Authentication-Oriented Approach to Model the Crosscutting Constraints in Sequence Diagram Using Aspect OCL

Ubaid Ullah,[1] **Usama Musharaf**,[2] **and Muhammad Haleem**[3]

[1]*Department of Computer Science and Information Technology, BRAINS Institute, Peshawar, Pakistan*
[2]*Department of Computer Science, FAST National University of Computer and Emerging Sciences, Peshawar, Pakistan*
[3]*Department of Computer Science, Kardan University, Parwan-e-Du Square, Kabul, Afghanistan*

Correspondence should be addressed to Muhammad Haleem; m.haleem@kardan.edu.af

Received 3 May 2022; Revised 20 November 2022; Accepted 24 November 2022; Published 16 December 2022

Academic Editor: Mahmood Niazi

The modeling of software functional requirements is very important in the development of software systems. In addition, it is also highly important to model security requirements, specifically authentication requirements, to prevent security risks and threats at the design level. The authentication requirements impose various constraints on granting access to only legitimate users to use computer resources. These constraints play an important role in the modeling of functional and authentication concerns of software systems. However, due to their crosscutting nature, their modeling results in pervasiveness across several design models. These constraints crosscut other constraints, which raises the problems of tangling and scattering. As a result, it is highly difficult to manage and maintain the constraints put on design models. Moreover, taking ad hoc approaches to deal with these constraints in complex systems is likely to result in faulty design models. All the existing approaches only deal with crosscutting behavior and completely ignore the crosscutting constraints. Therefore, our main research objective is to improve the modeling of crosscutting constraints and remove the scattering and tangling problems. For this purpose, we propose an authentication-oriented approach to modeling these constraints using the aspect-oriented technique. Using this approach, two case studies are implemented. Finally, the results show that the tangling and scattering problems are eliminated by separately modeling the crosscutting constraints as aspects using aspect OCL. This reduces the overall number of constraints and lowers the maintenance and management effort, which ultimately reduces the complexity of sequence models. The constraint-oriented sequence models are successfully verified, which shows that the output models are correct and complete. To conclude, our research approach is more useful and effective than the other approaches as it covers all the necessary steps required for functional and authentication behavior and constraint modeling.

1. Introduction

Modeling the functional requirements of a software system at the design stage is highly essential for supporting various software development activities. Moreover, software researchers have mostly concentrated on the modeling of functional requirements, but the modeling of non-functional requirements has been largely ignored [1, 2]. Among these non-functional requirements, authentication requirements are viewed as the most important [3] and should be addressed early in the software development process [4–6]. Authentication [7] is one of the software security aspects that refers to the confirmation and verification of a user's identity. It confirms that the data stored on a computer or network should be accessible only to authenticated users. Authentication requirements, from a design perspective, raise security challenges if they are not properly handled in a design. These requirements are considered crosscutting, which creates scattering and tangling problems. Therefore, researchers advise handling authentication behavior modeling carefully and separately from the software functional design during the design phase [4, 6, 8–10]. Apart from authentication behavior, authentication constraints also play a significant role in model-driven engineering [11, 12]. With UML models, constraints are used to make the models more accurate, consistent, and complete

according to the requirements specification [12–14]. Using the object constraint language (OCL), these constraints are normally applied to the design models for representing the system. But due to their crosscutting nature, these constraints also face scattering and tangling problems and become dispersed across several models [15], which make these models difficult to maintain and comprehend [16, 17], since they further increase the software complexity and result in highly chaotic and error-prone models. As a result, it is not beneficial for systems like banking systems, critical systems, etc. to model authentication constraints in this manner. Furthermore, taking an ad hoc approach to address these constraints for a large application is likely to result in faulty design models. These faulty models may result in security breaches or failures.

Hence, keeping all these problems in mind, we propose an authentication-oriented approach to provide a solution to the challenges of functional and security crosscutting constraints. This approach is based on aspect-oriented modeling and aspect OCL [15, 18] to identify and model the crosscutting behavior and constraints as aspects. This approach significantly reduces security vulnerabilities and defects, as well as the management and maintenance effort required for constraints, and also development time and cost [19, 20]. Moreover, the proposed approach reduces the burden on software developers by addressing security earlier. To implement the research, we used the UML class and sequence diagram for modeling. The class diagram is good at visualizing the software structure, and the concepts of aspect-oriented modeling are already embedded in the class diagram [1, 10]. As a result, many researchers used this diagram in security modeling [9, 10]. Another reason is that behavioral diagrams, such as sequence diagrams, are normally created from class diagrams to show the system's behavior. The sequence diagram is mostly used in industries and academia. This diagram is also used by researchers for security modeling to demonstrate security behavior [10]. Insecurities and weaknesses in security mechanisms can be easily identified with sequence diagrams [21].

Our research has made the following significant contributions:

(1) Our first contribution is the identification and separation of crosscutting constraints from simple OCL constraints at the design stage.

(2) Our second contribution is the extension of the aspect OCL for the specification of authentication constraints.

(3) Our third contribution is the modularization of the functional and authentication crosscutting constraints as constraint aspects using aspect OCL and weaving the constraint aspects into their respective sequence models.

(4) The fourth contribution is the verification of the constraint-oriented sequence model in terms of correctness and completeness.

The structure of the paper is as follows: In Sections 1 and 2, the introduction and related work are discussed. In Section 3, the proposed research approach is discussed. Section 4 is about constraint specification using OCL and Aspect OCL. Section 5 includes the implementation of the case studies, and Section 6 shows the results and significance. Section 7 concludes the research and future work.

2. Related Work

The related work shows that various authors contributed to the modularization of the crosscutting quality attributes using the aspect-oriented technique. Some authors also worked on the object constraint language (OCL) for specifying the constraints. Therefore, the related work is further divided into three subsections, i.e., the first one is related to modeling, the second one is related to OCL, and the third one is related to the overall summary and research gaps in the literature review.

2.1. Aspect-Oriented Modeling (AOM). Ullah et al. [10] proposed an aspect-oriented methodology to model the authentication crosscutting concerns as aspects in the mal sequence diagram. This methodology supports the UML extension mechanism and security profiles. The authors composed the authentication aspect with the application design to form a woven sequence diagram using matching and weaving mechanisms. The matching and weaving of the woven model are proved mathematically to check its completeness and soundness.

Ali et al. [1] modeled the robustness crosscutting behavior. The authors presented an aspect-oriented approach that supports the modeling of crosscutting behavior in the UML state machine diagram.

Mouheb et al. [6] presented a comprehensive methodology to support the design of security attributes. This methodology is based on an aspect-oriented technique and supports UML class and behavioral diagrams such as sequence, activity, and state machine diagrams.

Cooper et al. [22] presented a formal analysis and design framework (FADF) that is based on an aspect-oriented technique to provide a simple repository of aspects and support the visualization of these aspects in the class diagram.

Georg et al. [23] presented a class and sequence diagram using an aspect-oriented approach to design software systems that are highly secure and resistant to threats. The authors also conducted a risk assessment to identify potential software vulnerabilities and risks.

Sherief et al. [24] proposed a framework for threat-driven modeling to determine which threat requires mitigation and how to mitigate it. The authors mainly concentrated on the design level. Aspect-oriented stochastic Petri nets are used to specify the functions and threat mitigation mechanisms.

Xu and Nygard [25] presented a threat-driven approach to focus on the designing of functional and security mechanisms using aspect-oriented Petri nets to prevent integrity threats and ensure a secure software system. The complete security design is created by weaving the mitigation mechanism with the functional model.

Qiu and Zhang [26] proposed an aspect-oriented approach to address the maintenance problems of safety-critical systems. The redundancy tactics are localized as separate aspects from the functional models. These aspects are woven into the functional component diagram to form the application design. The results show that the proposed approach significantly reduces the rate of evolution of models and the number of model elements needed for design.

Ray et al. [9] proposed an aspect-oriented approach to address the crosscutting behavior modeling of access control concerns in UML diagrams. These concerns are separated from the software's primary model to reduce pervasiveness problems and strictly enforce security policies. Later, these concerns are woven into the primary model to form an application design. The results show that separately specifying the access control concerns eases the evolution of models.

Fan et al. [27] proposed a comprehensive aspect-oriented method for cloud computing to address resource management and scheduling issues. The authors only focused on the running time, processing, and reliability of resources. To construct the complete resource scheduling model, the authors first created the Petri net-based models for the base layer, meta-object layer, and other components, and then weaved these models into the main resource model. Results reveal that the proposed method has improved the efficiency of cloud computing.

Fuentes et al. [28] presented a process that is based on aspect orientation and executable modeling to overcome the problems of pervasiveness, reusability, and maintenance in context-aware systems. In this research, the aspect-oriented technique helps to modularize the context-aware crosscutting concerns into a separate module, while executable modeling supports the execution of the resultant models in different scenarios and conditions to reason about application design.

Rademacher et al. [29] proposed a method for streamlining the use of various technologies in microservices-based architectural software systems. This method offers numerous technology-related decisions in microservice architecture (MSA) and also provides justifications for these decisions using aspect-oriented modeling. As a result, a set of languages for model-driven microservice development is enhanced with capabilities for defining, modularizing, and using MSA technological elements.

Alhamad and Hassan [30] proposed a framework for security-based aspect-oriented models to address security concerns in intelligent systems. The unified modeling language (UML) is used to model and represent security aspects.

Yang and Wei-Dong [31] developed an approach for modeling the traversing features in concurrent software systems. The approach is based on the UML statechart diagram and aspect-oriented modeling (AOM). Using AOM, the traversing features and primary system functions are separated to provide benefits in terms of traceability, adaptability, and loose coupling.

Flotyński [32] proposed a method for visual aspect-oriented modeling of explorable XR environments. An explorable XR environment is one in which the behavior of users and 3D objects, including actions and interactions, is tracked and depicted using domain knowledge and visual descriptors.

2.2. Object Constraint Language (OCL). The work in the paper [33] is the only one discovered that uses OCL to provide role-based authorization constraints. This work assists system developers in understanding the requirements and restrictions for developing secure systems. The authors demonstrated how to create role-based authorization constraints using the industry-standard constraints specification language. They developed precondition constraints, separation of duties, and cardinality constraints before employing constraints specified by a formal language such as the role-based constraints language (RCL 2000). RCL 2000 [34] established a formal language with elements, syntax, and semantics for specifying role-based authorization constraints. The language featured two selection features and used role-based access control 96 (RBAC 96) [35].

Wang et al. [36] discussed details on how to use OCL to define constraints in RBAC. This paper has significantly contributed in three ways. To begin, this study greatly improves the work of [33] by discussing several constraint scenarios in the advanced role-based access control model and administrative role-based access control (ARBAC99) [37]. For access management, the authors provide different constraints that enable administrators to permit or revoke roles for users as mobile or immobile members.

2.3. Summary and Research Gap. The literature review shows that the crosscutting behavior of various quality attributes, such as context awareness, authentication, reliability, robustness, etc., is modeled using the aspect-oriented technique. A lot of work has also been done on OCL constraints. However, no solution to the authentication crosscutting constraints is presented by the authors. No work has been done on the modeling of the crosscutting constraints in UML diagrams, for instance, sequence diagrams. There is no approach offered to separate these constraints from non-crosscutting OCL constraints. Apart from that, verification of the constraint-oriented sequence diagram is not evident in the literature. As a result, the following research questions are developed to address the modeling of crosscutting constraints and the verification of constraint-oriented sequence diagrams using mathematical theorems:

RQ1. How to model the crosscutting constraints in a sequence diagram using aspect OCL?

RQ2. How to verify the correctness and completeness of a constraint-oriented sequence diagram?

3. Proposed Methodology

We propose an authentication-oriented modeling methodology that consists of three major steps, as shown in Figure 1. This methodology is based on the aspect-oriented

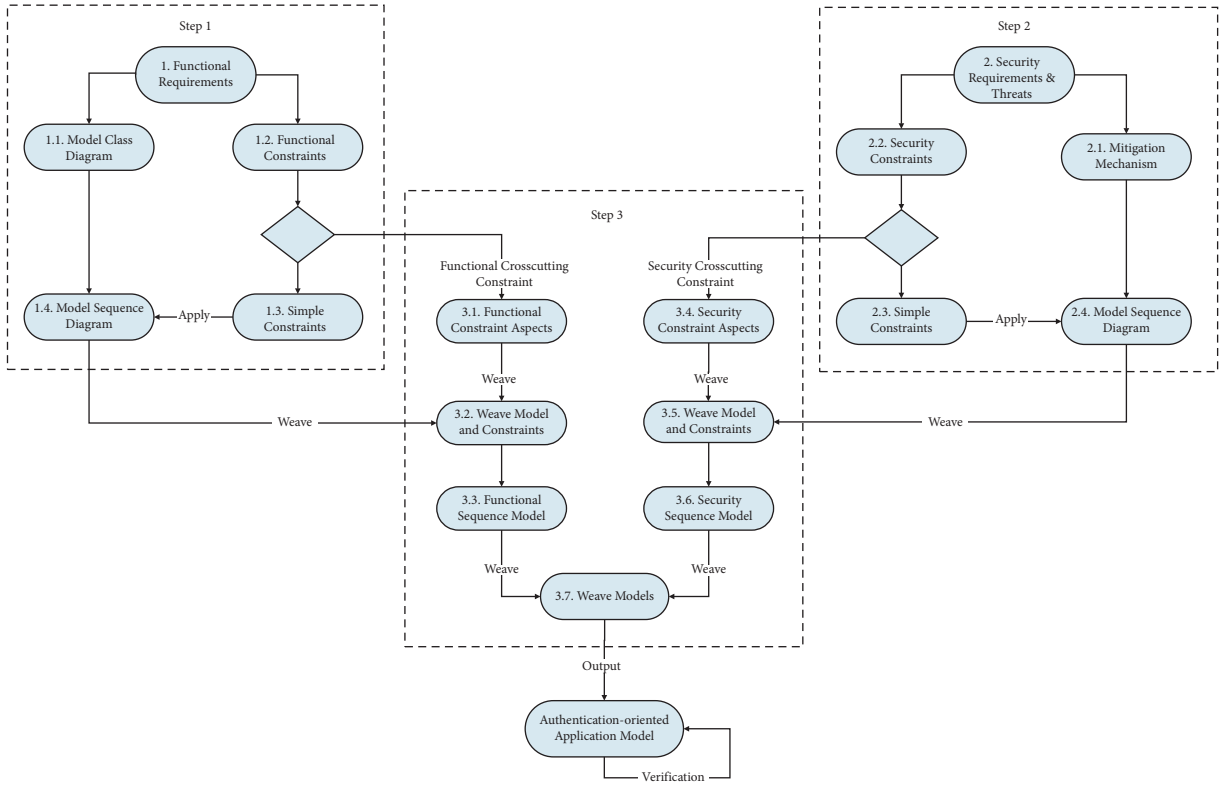

Figure 1: Authentication-oriented modeling approach.

technique and supports UML diagrams such as class and sequence diagrams for modeling the functional and security authentication behavior and constraints.

3.1. Functional Requirements. Step 1 involves the modeling of functional requirements in class and sequence diagrams and the separation of functional constraints into simple and crosscutting constraints. The simple constraints can be directly applied to the sequence diagrams, while the crosscutting constraints are specified separately as functional constraint aspects in step 3.1.

3.2. Security Requirements and Threats. In step 2, the approach takes security requirements as input for modeling. These requirements are extracted to provide a mitigation mechanism in step 2.1 and security constraints in step 2.2. The mitigation mechanism is modeled in a sequence diagram in step 2.4. The security constraints are further divided into simple constraints and crosscutting constraints. The simple constraints in step 2.3 can be directly applied to the sequence diagram, while the crosscutting constraints are modeled separately as security constraint aspects in step 3.4.

3.3. Weaving. In step 3, the functional constraint aspects are composed with the sequence diagrams to generate functional sequence models in step 3.3 using a weaving mechanism, while the security constraint aspects are composed with the authentication models to form security sequence models in step 3.6. The security sequence models are further composed with the functional sequence models to form authentication-oriented application models. These application models are further verified for correctness and completeness.

4. Crosscutting Constraints and Aspect-Oriented Technique

In this section, the crosscutting constraints and their specification using OCL and aspect OCL, and the features used in the aspect-oriented technique for modeling are discussed.

4.1. Object Constraint Language (OCL). The OCL is used to specify software model constraints [38]. In practice, OCL is utilized in industries [39, 40] to define constraints as expressions. Moreover, a significant benefit of this language is that it is not dependent on any programming level languages [41] and has no side effects [42, 43]. The OCL specifies simple constraints precisely [44], but it has no solution for crosscutting constraints to specify. A constraint is considered crosscutting if the same constraint exists in various operations in different classes. For example, in Figure 2, we have three classes, i.e., students, network administrators, and faculty, that have login operations. These classes have the same constraint for login operations, i.e., the constraint has the same invariant, precondition, and postcondition. Therefore, this type of constraint is an example of a crosscutting constraint.

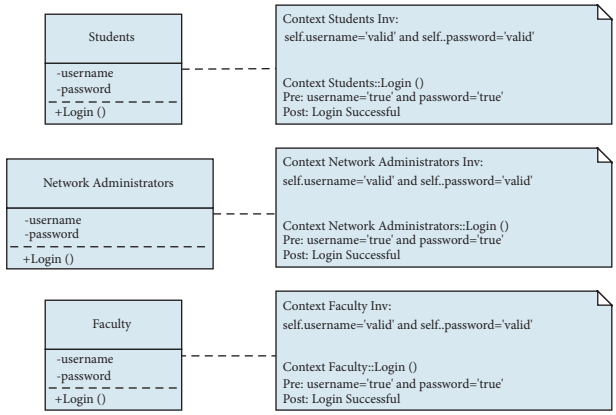

Figure 2: OCL constraints.

4.2. Aspect OCL. The aspect OCL specifies the crosscutting constraints as separate units called aspects [15, 18]. The aspect OCL provides advantages to eradicate the problems of scattering and tangling during the design. As shown in Figure 3, scattering means that the same constraint is present in multiple classes, whereas tangling means that the same operation or class localizes multiple constraints. It is time-wasting to write the same constraint again and again for the same operation in different classes. Therefore, to eliminate this problem of duplication, we have to write the constraint once and weave it with the operations where it is required, as we can see in Figure 4.

To specify the authentication crosscutting constraints using aspect OCL, we extended the aspect OCL constraint as shown in Figure 5. The authentication constraint has the following conditions:

Invariant condition. An invariant is a condition that must remain true for all instances of a class.

Precondition. A precondition is a condition that must be true before the execution of an operation in a class.

Postcondition. A postcondition is a condition that must be true after the execution of an operation in a class.

Body condition. A body condition is a condition that holds the result of an operation.

4.3. Aspect-Oriented Modeling (AOM) Constructs. The AOM constructs play a vital role in the modularization of aspects. These constructs are used to modularize the behavioral aspects and crosscutting constraints.

4.3.1. Aspect. Aspect modularizes the crosscutting behavior [1, 6, 10] and crosscutting constraints [15] into separate units. Aspect is separately managed and maintained from functional models and is composed with the models at specified join-points using pointcut expressions. In the case of the constraint specification, the aspect OCL divides the constraint aspect into two parts [15], i.e., mapping and aspect. The mapping part is used to specify the target elements to which the constraint aspect is to be applied. The target elements can be UML classes or operations. The aspect part is used to define a pointcut. The pointcut includes the introduction or advice [15] to insert the constraint. In Figure 4, the mapping is defined across several classes, while the aspect has defined a pointcut, i.e., unique Usernames to insert this constraint.

4.3.2. Introduction. The introduction is used to introduce a new constraint on a model element if there is already no constraint defined. The introduction inserts a constraint at the join point chosen by the pointcut. The aspect OCL presents various types of introduction for constraint specification [15].

4.3.3. Advice. The advice is used to determine how the aspects are inserted at the join point(s) selected by the pointcut. In the case of aspect OCL, various types of advice [15] are used to add a constraint to already defined constraints on a model element.

4.3.4. Join point. The point in the design model where the aspect is woven to add behavior.

4.3.5. Pointcut. It is used to select the model's join points to apply the aspect. The pointcut acts like a selection query [15]. Figure 6 shows the syntax of the pointcut.

4.3.6. Weaving. Weaving is a technique for incorporating behavioral or constraint aspects into models.

4.3.7. Let. Let is used in the constraint aspect to define a mapping among classes, operations, and properties. For example, in Figure 4, the expression is used for defining classes, i.e., students, network administrators, and faculty, that include the crosscutting constraints.

5. Analysis

Our main objective is to improve software modeling by overcoming the problems of functional and authentication crosscutting constraints using aspect OCL in sequence diagrams and to verify the constraints-oriented design models. This objective is accomplished using two research questions. The first research question (RQ1) is about the modeling of the crosscutting constraints, and the second research question (RQ2) is about the verification of the constraint-oriented model. Two case studies, i.e., the ATM [45] and e-commerce [46], are implemented using the proposed approach to solve RQ1.

RQ1. How to model the crosscutting constraints in a sequence diagram using aspect OCL?

5.1. ATM Case Study

5.1.1. Functional Requirements

(1) Model a Class Diagram. In this step, a class diagram for the ATM system is created using the functional requirements as shown in Figure 7. The goal of this diagram is to

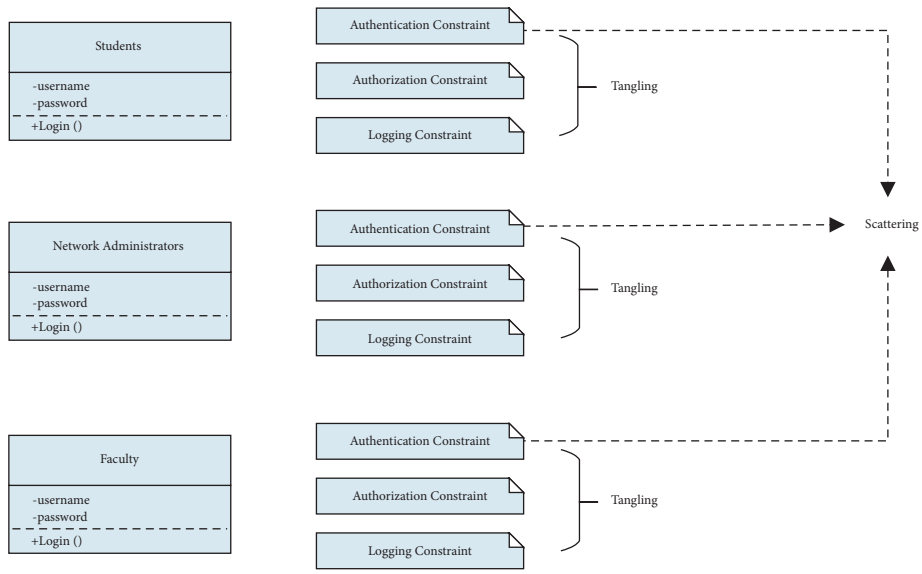

Figure 3: Tangling and scattering.

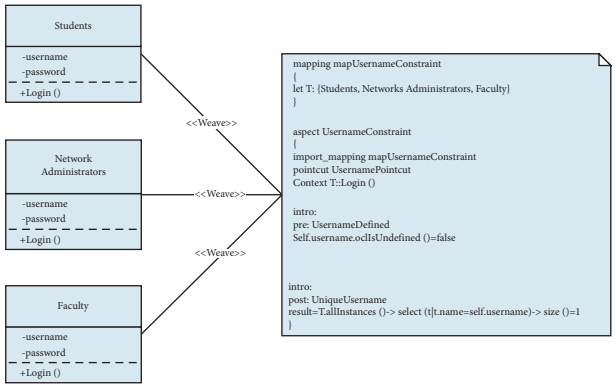

Figure 4: Constraint aspect.

depict the structural perspective of the ATM and how different objects interact with one another to accomplish the ATM's functionalities.

(2) Extract Functional Constraints. In this step, the functional constraints (*Fc*) are extracted from the functional requirements. These constraints are further divided into simple and crosscutting constraints. The extracted constraints for the ATM system are listed in Table 1.

(3) Simple Constraints. In this step, the simple constraints, i.e., the non-crosscutting constraints, are applied directly to the sequence diagram. In the current situation, we have no simple constraints.

(4) Creation of a Sequence Diagram. A sequence diagram is used to depict the sequence of actions of a software system, and a class diagram is normally used in the modeling of this diagram. Figures 8–10 show the sequence diagrams for withdrawing, depositing, and transferring cash, respectively.

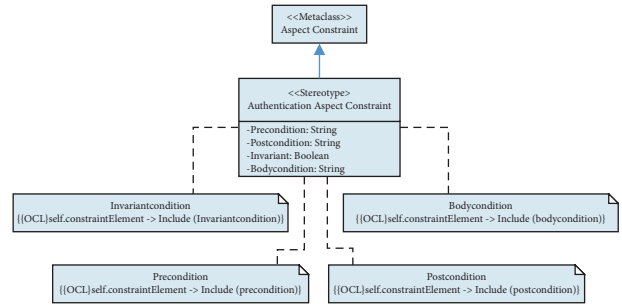

Figure 5: Authentication constraint aspect.

pointcut <name>
Context <class or method name where the aspect apply>
<Expression>
<advice | introduction>

Figure 6: Pointcut.

5.1.2. Security Requirements and Threats Identification. The most important part of software modeling is to consider security during the early stages of software development. Software security may be jeopardized if security threats are not considered and mitigated. In the case of the ATM system, two threats [47–49] are taken to understand how the hacker threatened ATM security. These threats are given in Figure 11.

ATM Card Skimming. Card skimming threats involve stealing user card data by using a device called a "skimmer," which looks like a scanner. This device can correctly scan and record the card's critical data when it is entirely positioned over an ATM's card slot region, The hacker uses this information to conduct transactions [50]. The hacker also uses a camera to easily record the PIN typed by the user [51].

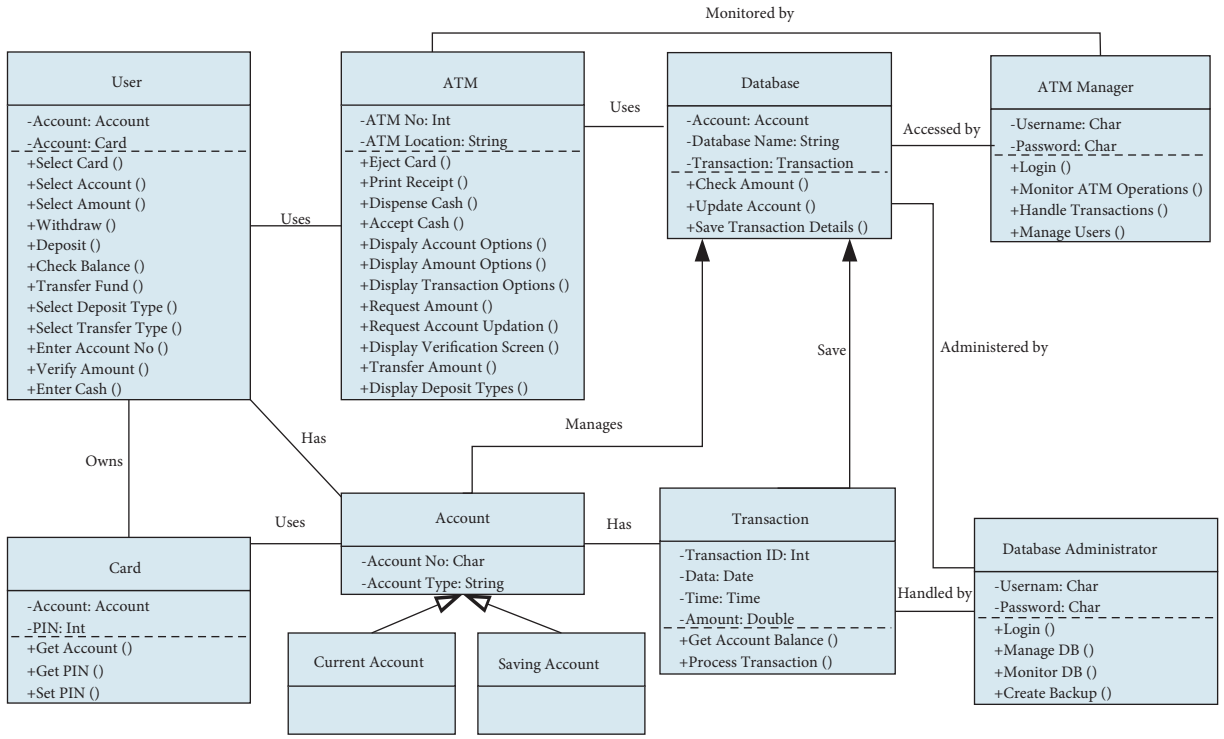

Figure 7: Class diagram.

Table 1: Functional constraints.

Functional constraint	Name	Description
*Fc*1	Transaction amount limit for transfers and withdrawals	The transaction amount must be greater than or equal to 500 and less than or equal to 50,000.
*Fc*2	Account updating in the event of a withdrawal or transfer	If the transaction is successful, the balance must be deducted from the account and the account must be updated.
*Fc*3	Account updating in the case of a deposit	If the transaction is successful, the balance must be deposited into the account and the account must be updated.
*Fc*4	Save transaction details	The transaction information must be saved.
*Fc*5	Successful transaction	The transaction must be successful if the ATM withdraws cash, accepts cash, and transfers cash.

ATM Card Trapping. Card trapping attacks allow a hacker to capture an ATM card using a special device. In this attack, the camera is also utilized.

(1) Extract Mitigation Mechanism. At this point, the preventive strategy for mitigating threats is extracted. The preventative strategy of multi-factor authentication (MFA) is a sort of authentication strategy that ensures that the system is only accessible to genuine users. It ensures that only legitimate users are permitted to access the system, and users who are not verified by the system are prohibited. Several techniques [52] are used to certify the user's validity and protect the user's data from being accessed by a hacker [53]. We believe that using a fingerprint as the third step of authentication will raise security to a greater degree. The MFA steps are described below.

Step 1: The ATM card is used to authenticate the user.

Step 2: The user's PIN is used for authentication.

Step 3: The user's fingerprint is used for verification.

(2) Extract Security Constraints. The security constraints (*Sc*) are extracted from the security requirements at this stage to impose restrictions and prevent security threats. These constraints are further divided into simple and crosscutting authentication constraints. The extracted constraints for the ATM system are listed in Table 2.

(3) Simple Constraints. The non-crosscutting authentication constraints can be applied directly to the sequence diagram in step 5.4. However, in the current scenario, we do not have simple constraints.

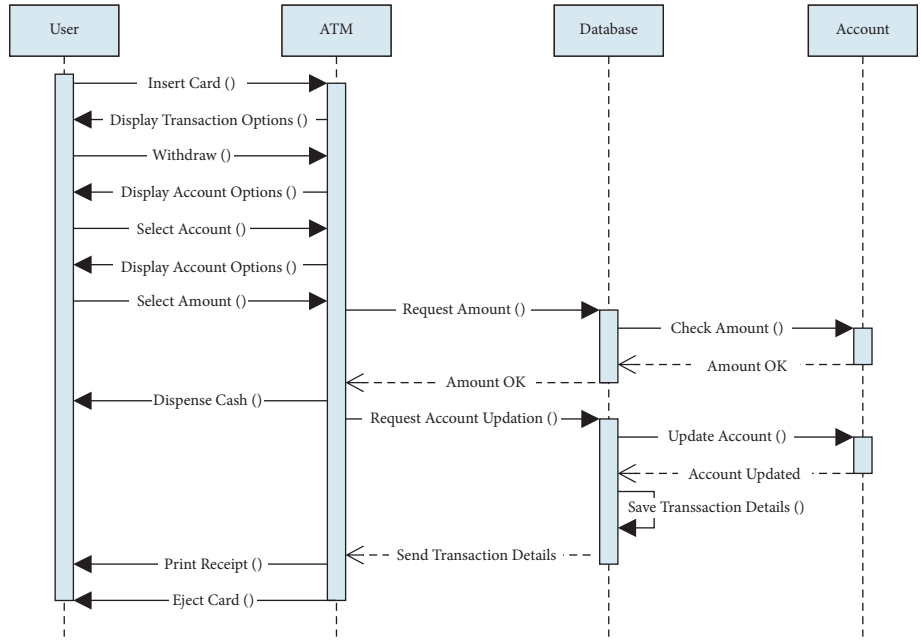

Figure 8: Withdraw model.

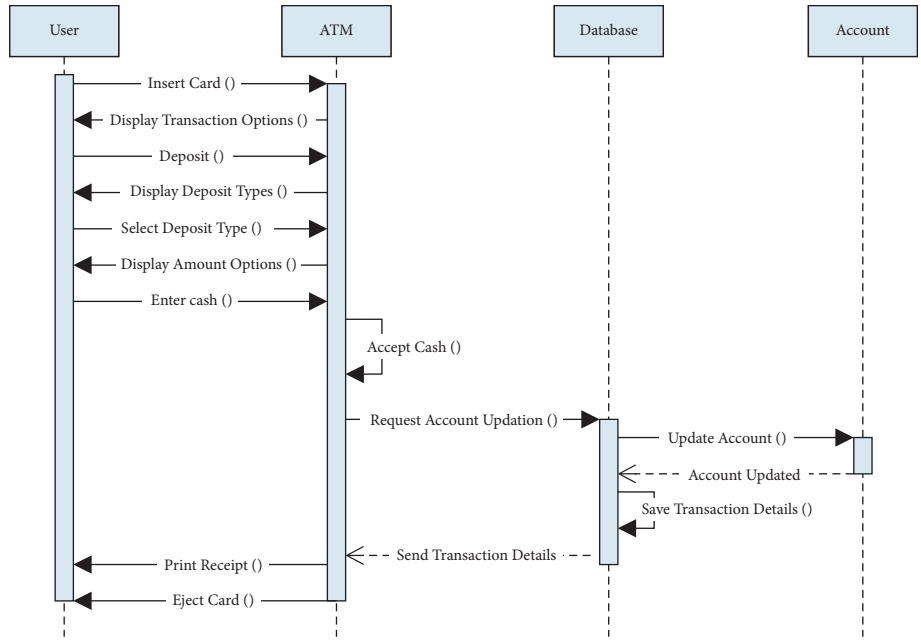

Figure 9: Deposit model.

(4) Model a Sequence Diagram. The prevention mechanism to prevent authentication threats is modeled using a sequence diagram as depicted in Figure 12. This diagram is known as the "authentication aspect" because it modularizes the authentication behavior as an aspect.

5.1.3. Weaving

(1) Functional Constraints Aspects. The constraints that are identified in step 5.1.2 are all crosscutting and must be specified separately as aspects using aspect OCL. The ATM functional constraint aspects are given below in Figure 13.

(2) Weave Functional Models and Functional Constraints Aspects. In this step, all the functional crosscutting constraints are weaved into the ATM functional models. The weaving lines for each constraint aspect are colored differently to differentiate from each other as shown in Figure 14.

(3) Functional Sequence Model. The resultant functional model for withdrawal balance is given below in Figure 15. We are using ref to indicate the reference to the actual functional constraints that are weaved at the join points in step 5.3.2. Similarly, we can obtain a deposit and transfer fund models after weaving.

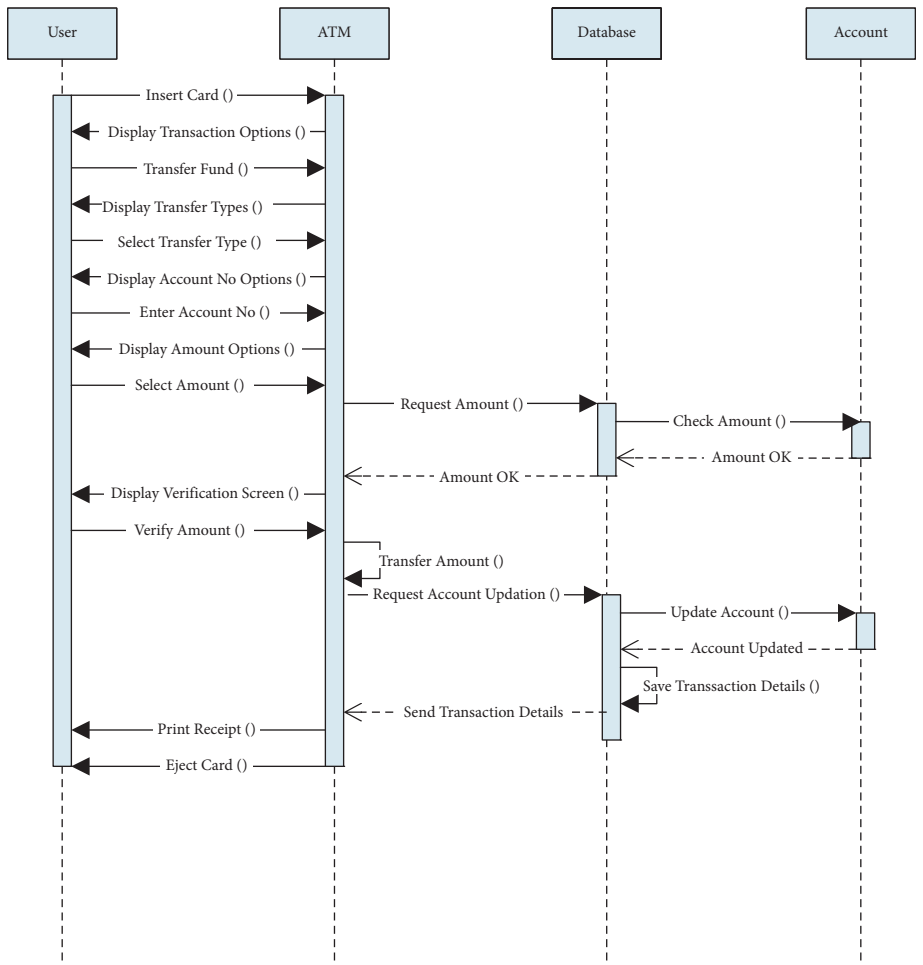

Figure 10: Transfer fund model.

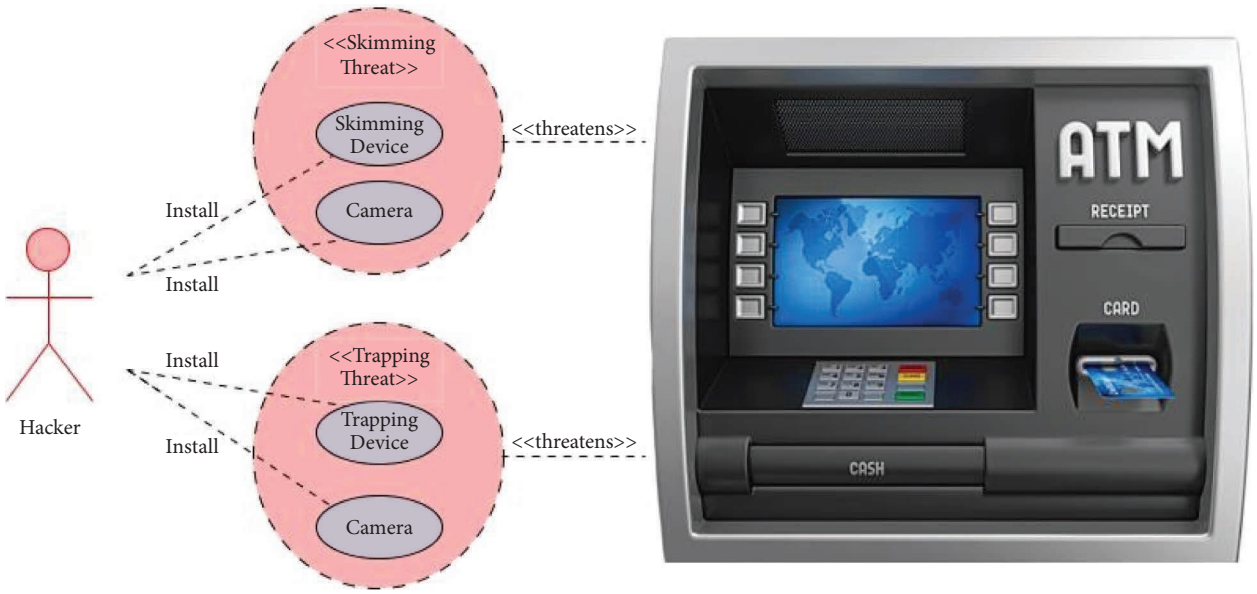

Figure 11: ATM threats.

Table 2: Security constraints.

Security constraints	Description
*Sc*1	The card number must be valid
*Sc*2	The account must be verified
*Sc*3	The PIN must be valid
*Sc*4	The attempts for PIN must be less than 4
*Sc*5	The fingerprint must be matched

(4) Security Constraints Aspects. The crosscutting security constraints are separately specified as security constraint aspects. For simplicity, we have considered only five constraints as shown in Figure 16.

(5) Weaving Security Model and Security Constraint Aspects. In this step, the security crosscutting constraints that are modeled separately are now weaved into the security authentication aspect to form a security sequence model in Figure 17.

(6) Security Sequence Model. The resultant authentication sequence model is given below in Figure 18. The ref indicates the reference to the actual constraint aspects that are weaved at the desired join points.

(7) Weave Models. In this step, the authentication model in step 3.6 is weaved into the functional models in step 3.3 to form an authentication-oriented application model as shown in Figure 19. The authentication design model is weaved at those join points where it is needed. The authentication-oriented model is also referred to as the "constraint-oriented sequence model." Similarly, we can get the authentication-oriented deposit and fund transfer models.

If a user intends to do a transaction such as withdraw, deposit, or transfer money, the user will insert a card. The ATM will direct the functional model to the authentication model to verify the user credentials, and following successful authentication, the ATM will start a sequence-wise transaction for the user. Apart from that, all the constraints must be fulfilled for a successful transaction to be made.

5.2. E-Commerce System Case Study. Using the proposed approach, the following authentication-oriented models for the e-commerce system are created in Figures 20–22. Instead of showing all the diagrams, we have only shown the output sequence models for three scenarios.

5.2.1. Verification of Constraint-Oriented Models. In this section, RQ2 is answered. We want to verify the constraint-oriented sequence models through mathematical theorems. These theorems verify that if the weaving process is correct and complete, then the constraint-oriented models will also be correct and complete. These theorems have previously been used to validate activity diagrams [4, 6] and sequence diagrams [10]. Moreover, the syntax and semantic rules for the weaving process are also provided for the sequence diagrams [10]. Therefore, we extended these theorems to verify the constraint-oriented sequence models. For simplicity, we have taken an authentication-oriented withdraw model in Figure 19 for the verification process. Similarly, we can verify other output models.

RQ2. How to verify the correctness and completeness of constraint-oriented sequence diagrams?

(1) Correctness of Constraint-Oriented Model. To verify the correctness of the constraint-oriented withdraw model, we have to prove expression (1). This expression is based on the constructs that contributed to the weaving process, i.e., the functional sequence diagram "*S*," authentication aspect "*a*," constraint aspect "*c*," and joinpoint "*j*," where *j* is the joinpoint in the authentication-oriented withdraw model. All these constructs are included in the following expression.

$$\text{"If Weaving}(S, a, c, j) = S'' \text{ then } (S, a, c, j, \text{weaving}) \longrightarrow (S'', a', c', j'', \text{end})\text{"}. \quad (1)$$

$(S, a, c, j,$ weaving$)$ represents the functional sequence diagram before weaving, while $(S'', a', c', j'',$ end$)$" represents the functional sequence diagram after weaving.

(1) Base Case

The base case includes the following conditions for the weaving process.

(1) Before weaving $(S, j) = S$
(2) Before weaving $(a) = a$
(3) Before weaving $(c) = c$

Using the conditions, the base case for weaving transformation becomes.

$(S, a, c, j,$ weaving$) \longrightarrow (S, a, c, j,$ end$)$.

(2) Induction Hypothesis

Using the hypothesis let $a = a'$, $c = c'$, the equation (1) becomes.

$$\text{"If Weaving}(S, a', c', j) = S'' \text{ then } (S, a', c', j, \text{weaving}) \longrightarrow (S'', a', c', j'', \text{end})\text{"}. \quad (2)$$

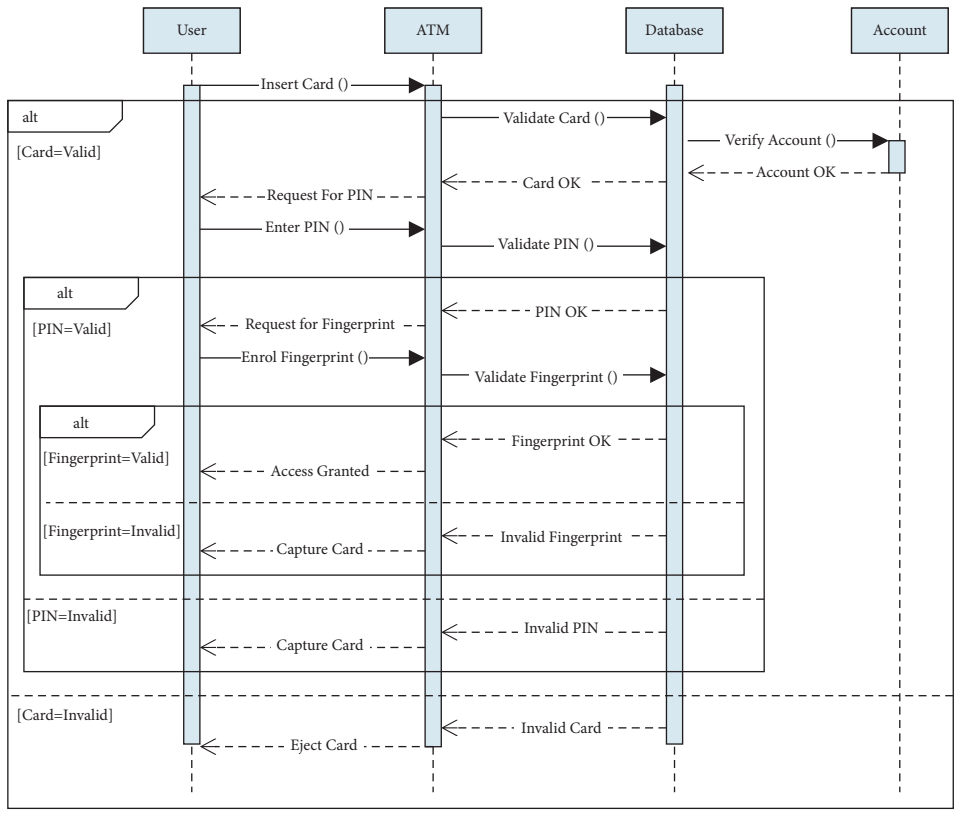

Figure 12: Authentication aspect.

Figure 13: Functional constraints aspects.

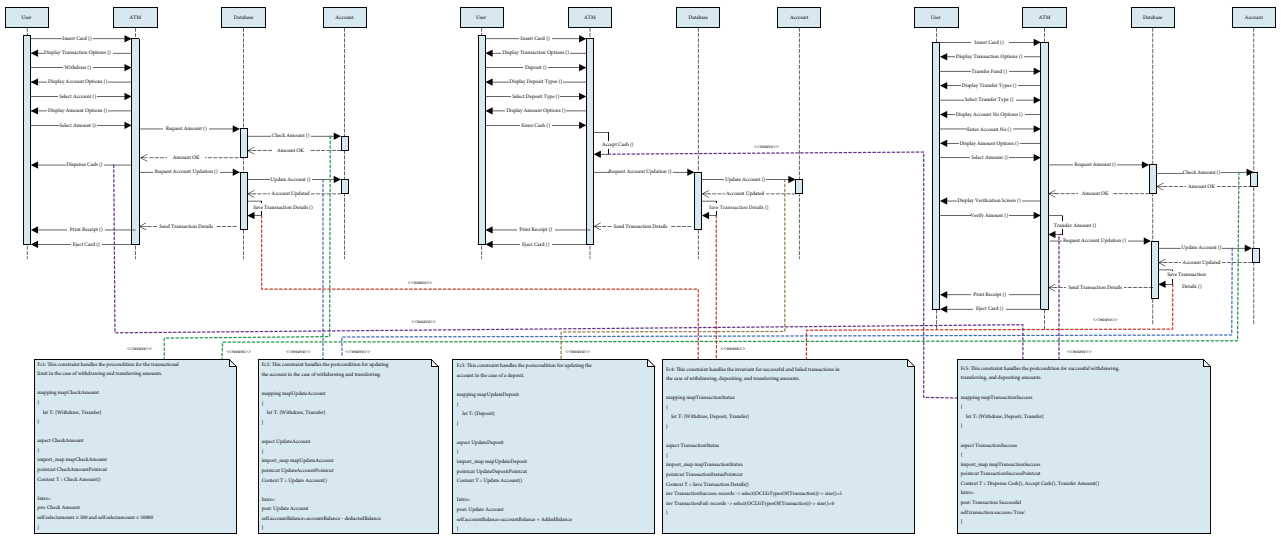

Figure 14: Weaving.

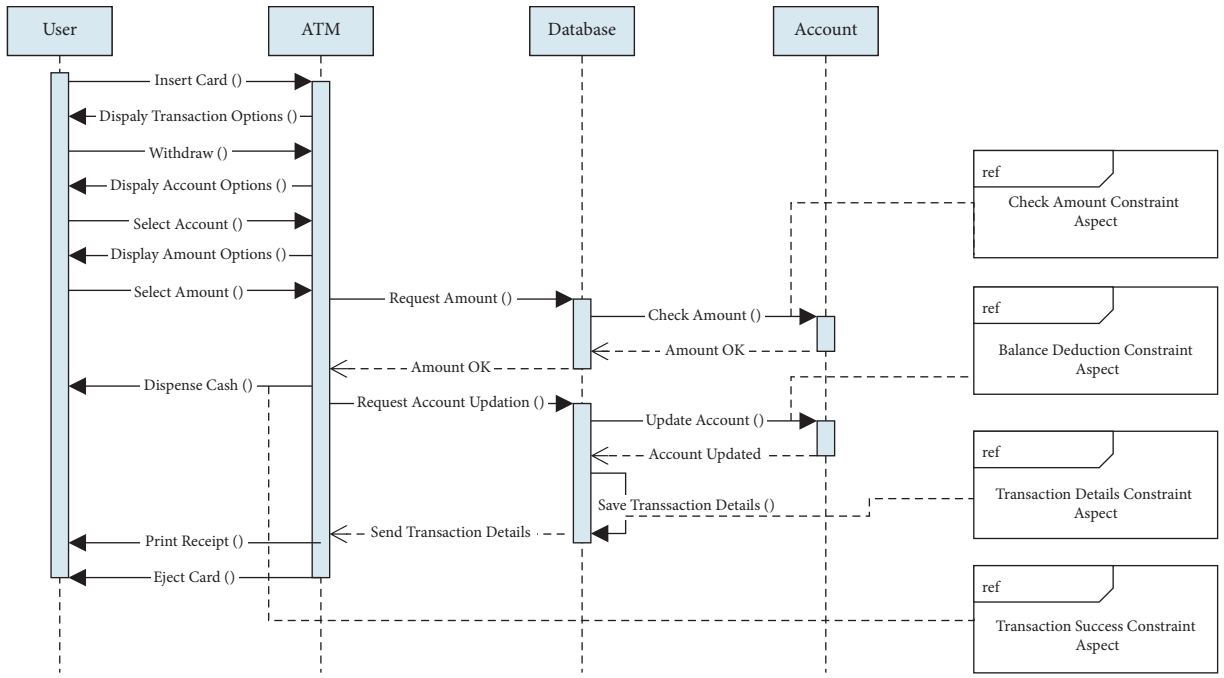

Figure 15: Withdraw model with functional constraints.

We used the following steps for the weaving process.

(1) We added the constraint aspects and authentication aspects. Hence, the addition can be written as. add.kind = introduction
(2) j is a type of join point: j.type ∈ Join point
(3) j belongs to the functional sequence diagram's join points: $j = S$. Join points

After the weaving process, when we apply the authentication aspects and constraint aspects, the transformation equation becomes

$$\text{“}(S, a, c, j, \text{weaving}) \longrightarrow (S', a', c', j', \text{weaving})\text{”}. \quad (3)$$

Using the hypothesis we can conclude that

$$\text{“}(S', a', c', j', \text{weaving}) \longrightarrow (S'', a', c', j'', \text{end})\text{”}. \quad (4)$$

Using the relation of transitivity for (3) and (4) which results in.

$$\text{“}(\mathbf{S}, \mathbf{a}, \mathbf{c}, \mathbf{j}, \mathbf{weaving}) \longrightarrow (\mathbf{S''}, \mathbf{a'}, \mathbf{c'}, \mathbf{j''}, \mathbf{end})\text{“} \quad (\mathbf{Proved}). \quad (5)$$

Hence, equation (1) is proved. We can conclude that the authentication-oriented withdraw model in Figure 19 is correct. The equation shows the correct transformation of the weaving process from (S, a, c, j, weaving) to (S'', a', c', j'', end). Moreover, it shows that we correctly weaved the aspects with the functional model.

```
Sc1: This constraint handles the precondition for card validation.

mapping mapValidateCard
{
letT: {Authentication}
}

aspect ValidateCard
{
import_map mapValidateCard
pointcut ValidateCardPointcut
Context T :: Validate Card ()

Intro::
pre: Card Validation
if self.IsCardIdOK then
 Valid
else
 Invalid
endif
}
```

(a)

```
Sc2: This contraint handles the precondition for account
verification.

mapping mapAccountVerification
{
 letT: {Authentication}
}

aspect VerifyAccount
{
import_map mapAccountVerification
pointcut VerifyAccountPointcut
Context T :: Verify Account ()

Intro::
pre: Account Verification
self.accountNo=CardId
}
```

(b)

```
Sc3: This constraint handles the precondition for PIN Validation.

mapping mapValidatePIN
{
 letT: {Authentication}
}

aspect ValidatePIN
{
import_map mapValidatePIN
pointcut ValidatePINPointcut
Context T :: Validate PIN ()

Intro::
pre: PIN Validation
if self.IsPINOK then
 valid
else
 Invalid
endif
}
```

(c)

```
Sc4: This constraint handles the precondition for PIN Attempts

mapping mapPINAttempts
{
 letT: {Authentication}
}

aspect PINAttempts
{
import_map mapPINAttempts
pointcut PINAttemptsPointcut
Context T :: Enter PIN ()

Intro::
pre: PIN Attempts
self.attempts <4 and self.attempts=attempts + 1
}
```

(d)

Figure 16: Continued.

```
Sc5: This constraint handles the precondition for fingerprint validation

mapping mapValidateFingerprint
{
 letT: {Authentication}
}

aspect ValidateFingerprint
{
import_map mapValidateFingerprint
pointcut ValidateFingerprintPointcut
Contect T :: Validate Fingerprint ()

Intro::
pre: Fingerprint Validation
if self.IsFingerprintMatched then
 Valid
else
 Invalid
end if
}
```

(e)

Figure 16: Authentication constraint aspects. (a) Card validation constraint aspect. (b) Account verification constraint aspect. (c) PIN validation constraint aspect. (d) PIN attempts constraint aspect. (e) Fingerprint validation constraint aspect.

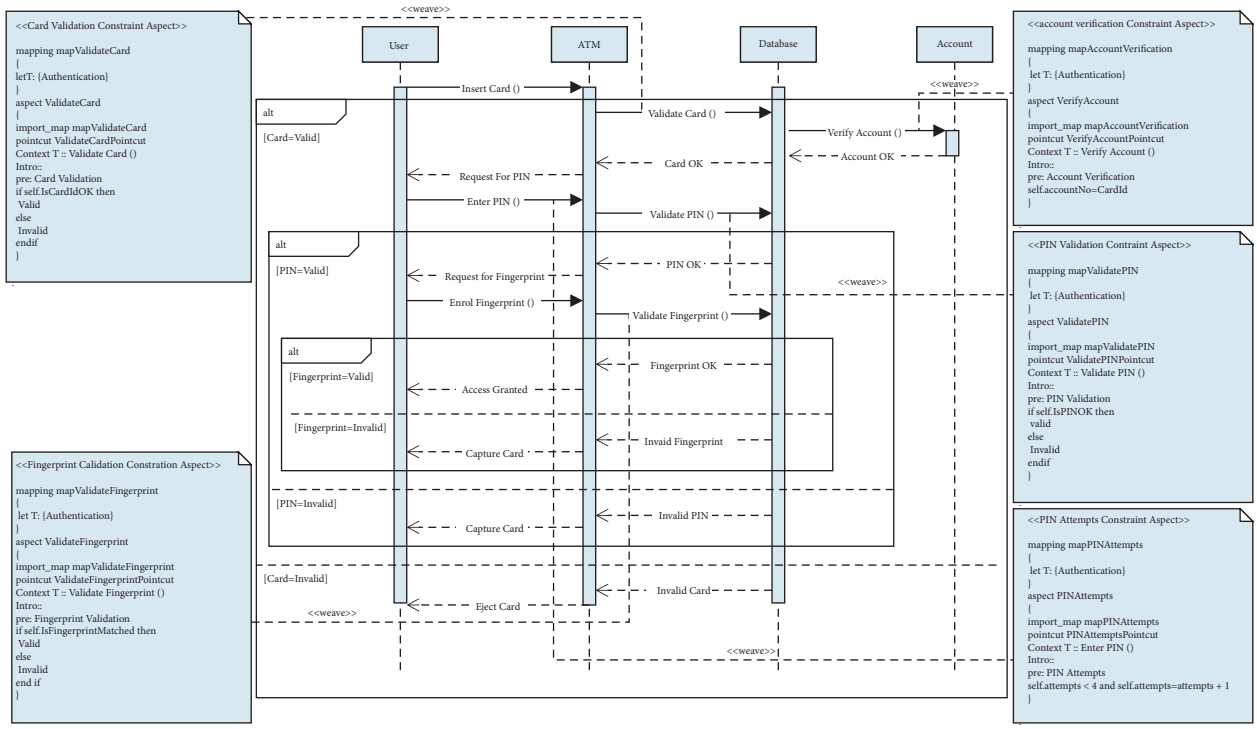

Figure 17: Weaving security model and constraints.

(2) Completeness of Constraint-Oriented Model. To verify the completeness of the constraint-oriented withdraw model, we have to prove expression (6). This expression is based on the constructs that contributed to the weaving process. The constructs are functional sequence diagram "*S*," authentication aspect "*a*," constraint aspect "*c*,", and a joinpoint "*j*,"

where j is the joinpoints in the authentication-oriented withdraw model. If the weaving is complete, then the output model will also be complete.

$$\text{“If Weaving}(S, a, c, j, \text{weaving}) \longrightarrow (S'', a', c', j', \text{end}) \text{then Weaving}(S, a, c, j) = S''\text{“}. \quad (6)$$

In expression (6), (S, a, c, j, weaving) represents before weaving, (S'', a', c', j'', end) represents after weaving, and (S, a, c, j) = S'' represents the complete diagram in Figure 19.

(1) Basis case

For the base case, the transformation expression using the end weaving can be written as.

“(S, a, c, j, weaving) ⟶ (S, a', c', j, end)”

But we know that (S, a, c, j, weaving) = S, therefore the expression becomes:

Weaving (S, a', c', j) = S.

(2) Induction Hypothesis

Using the hypothesis, let $a = a'$, $c = c'$, equation (6) becomes.

$$\text{“If Weaving}(S, a', c', j, \text{weaving}) \longrightarrow (S'', a', c', j'', \text{end}) \text{then Weaving}(S, a', c', j) = S''\text{“}. \quad (7)$$

As we know, the weaving process follows the following three steps.

(1) Our add adaptation can be given as add.kind = introduction.
(2) j has a type that belongs to the join point: p.type ∈ Join point
(3) j belongs to the sequence diagram's join points: $j = S$.Join points

Using all three steps for the weaving process, we can conclude that.

$$\text{Weaving}(S, a, c, j) = \text{Weaving}(S', a', c', j'). \quad (8)$$

We know that in expression (8).

$$\text{Weaving}(S', a', c', j')'' = (S'', a', c', j'', \text{end}). \quad (9)$$

We also know that.

$$(S'', a', c', j'', \text{end}) = S'' \quad (10)$$

Transitivity for (8) and (10) results.

$$\text{weaving}(S', a', c', j') = S''. \quad (11)$$

But according to equation (8), (S', a', c', j')” = weaving (S, a, c, j), therefore equation (11) becomes.

$$\text{Weaving}(S, a, c, j) = S'' \quad \text{(Proved)}. \quad (12)$$

Hence, equation (6) is proved. We can conclude from equation (6) that the authentication-oriented withdraw model is complete because if we take a look at this model in Figure 19, it contains a functional sequence diagram, an authentication aspect, constraint aspects, and all other join points, which indicate that all the constructs are same before and after weaving. Hence, our constraint-oriented model (S'') is proved in terms of completeness. Similarly, we can verify other constraint-oriented models.

6. Results and Significance

6.1. Results. The results that are concluded from the implementation of the case studies and the verification process are given below.

(1) Using the proposed approach, all the functional sequence diagrams of the ATM system in Figures 8–10 and their functional crosscutting constraints in Figure 13 are successfully modeled separately. Similarly, the security aspect model in Figure 12 and all its security crosscutting constraint aspects in Figure 18 are also modeled separately. These constraint aspects are now managed and maintained separately from the design models. The tangling and scattering are removed, as there is no need to write the same constraint again and again. We have to write the constraints once and weave them into the desired sequence models at their specified join points. In this way, the duplication of constraints is completely removed from sequence diagrams. The same results are true for the second case study.
(2) The overall constraints are reduced. In the case of the ATM case study, if we take both techniques, i.e., OCL and aspect OCL, for constraints specification, we can conclude that using OCL, the total constraints required for all functional diagrams are 26 while using aspect OCL, the same constraints are reduced to 10. Constraint specifications using both techniques are

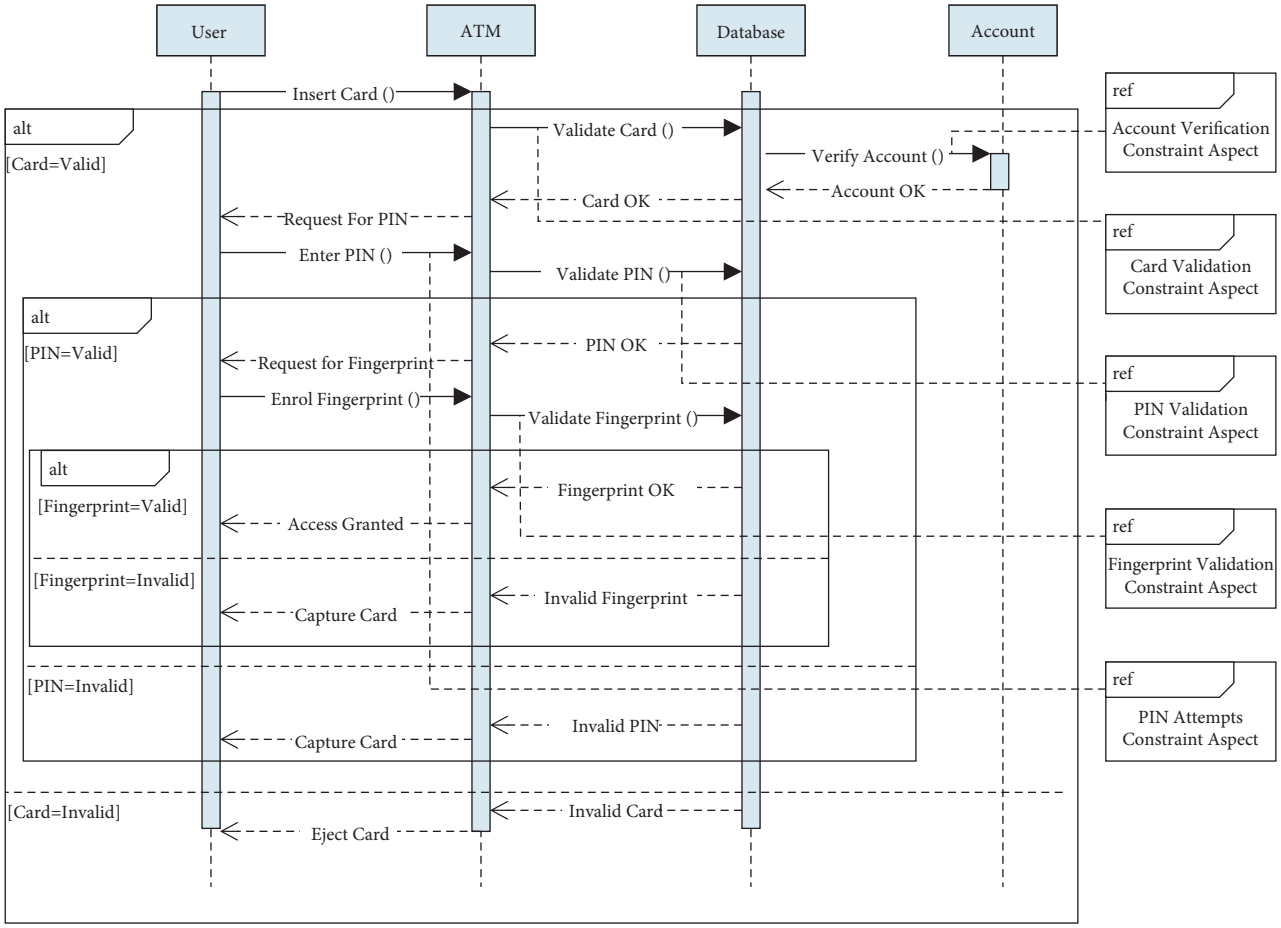

Figure 18: Authentication sequence model with constraints.

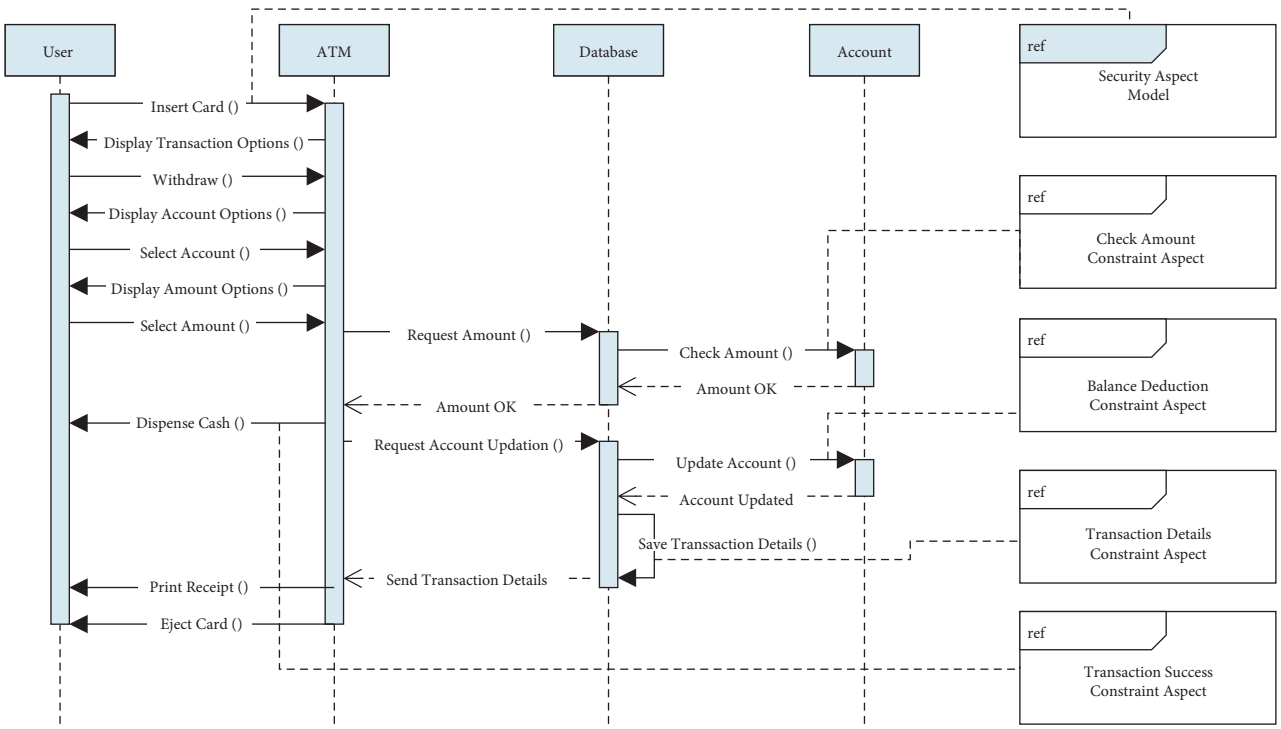

Figure 19: Authentication-oriented withdraw model.

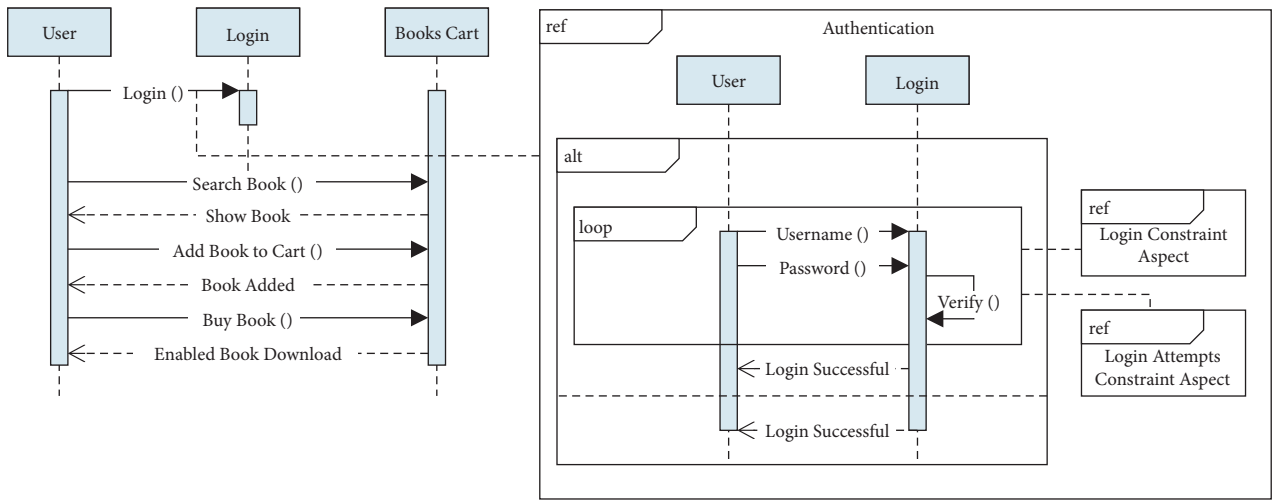

Figure 20: Authentication-oriented purchase model.

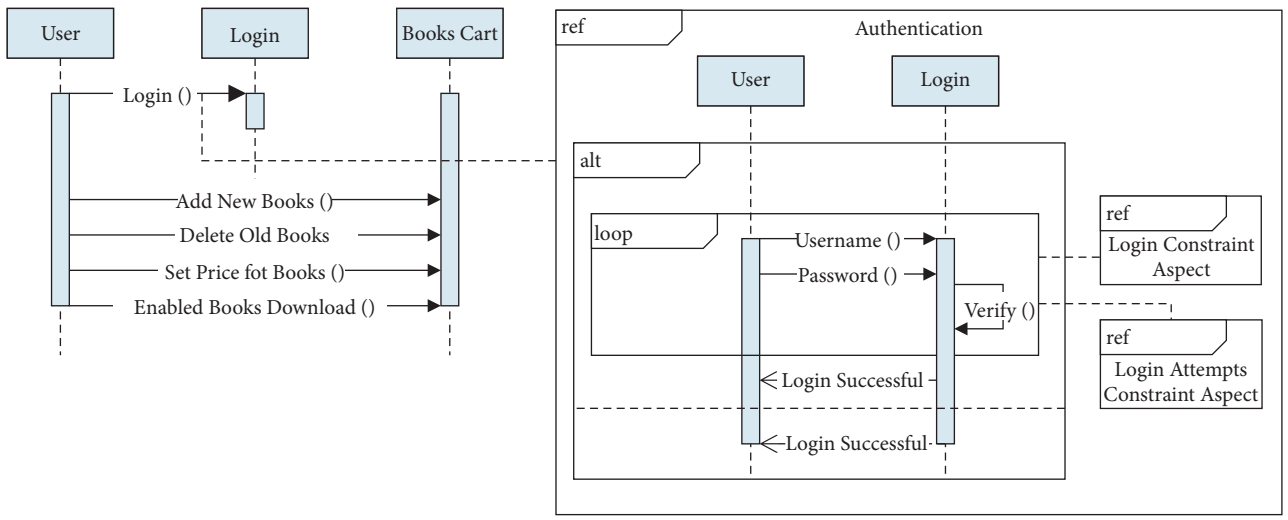

Figure 21: Authentication-oriented inventory model.

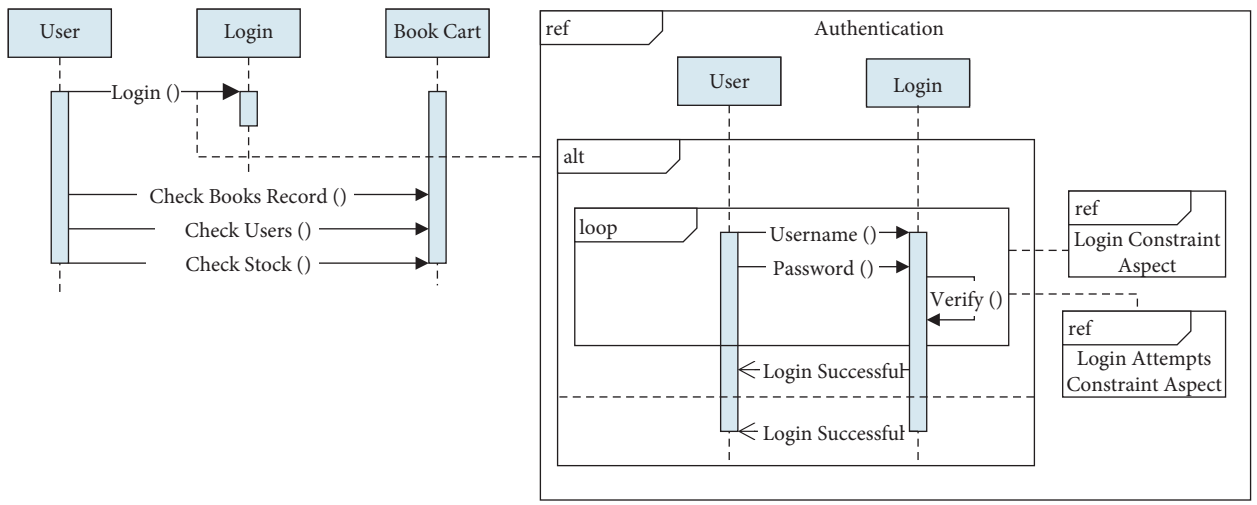

Figure 22: Authentication-oriented stock model.

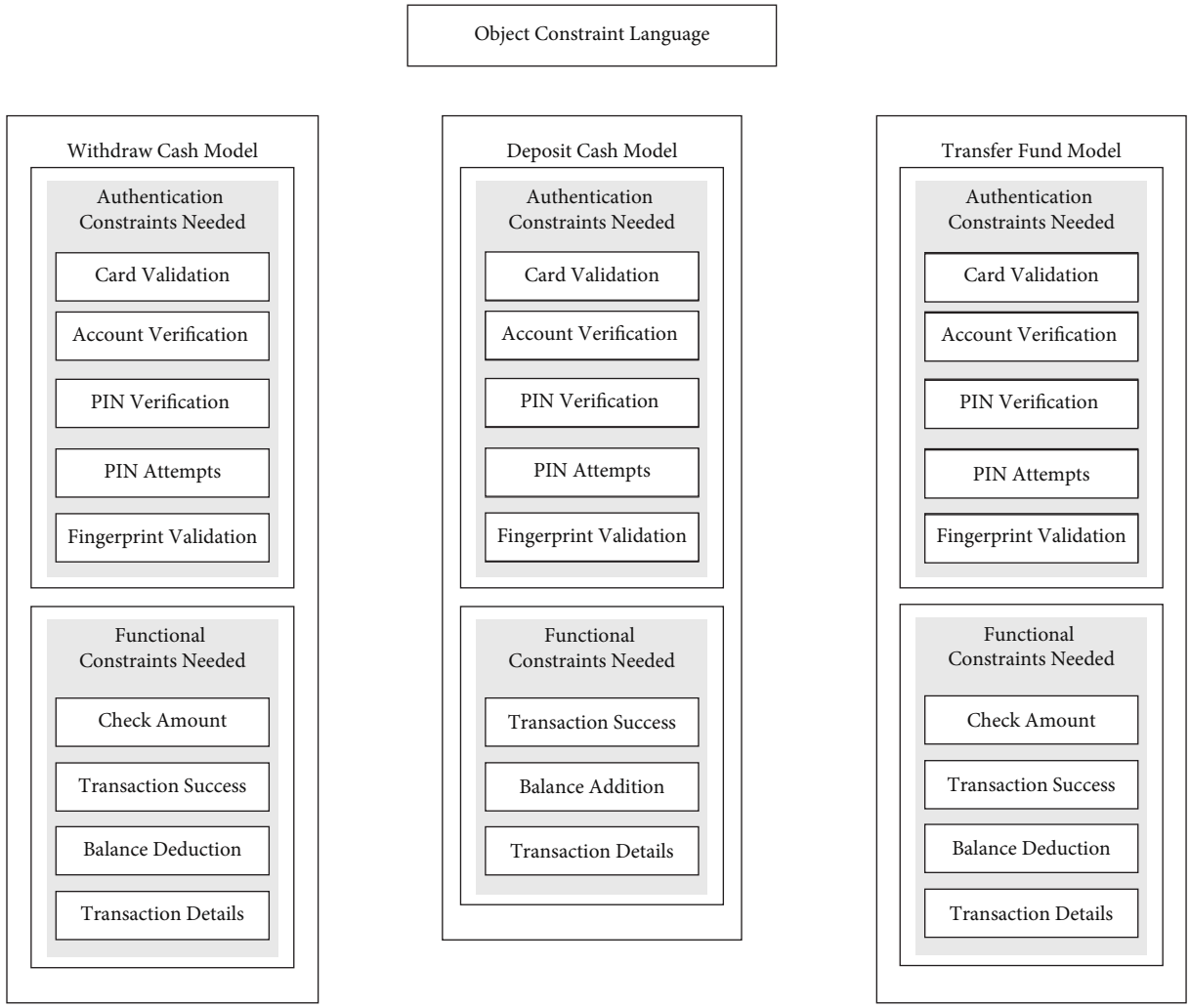

Figure 23: Constraints specification using OCL.

given below in Figures 23 and 24. Similarly, in the case of the e-commerce case study, all three functional models need 6 constraints, i.e., two constraints for each model. As a result of using aspect OCL, the constraints are reduced to two.

(3) The authentication constraint-oriented model has been successfully verified in terms of correctness and completeness, which shows that the output models are correct and complete.

(4) According to our findings, our approach is more effective and useful than the other approaches. Our methodology covers all the necessary steps required for behavior and constraint modeling using the aspect-oriented technique. Table 3 presents all the approaches discussed in the literature review. These approaches ignore the most important steps that are required for modeling the constraints using the aspect-oriented technique.

6.2. Research Significance. The proposed approach and its results are examined from various angles to determine whether they address modeling issues such as constraint reduction, separation of constraints, constraint evolution, complexity reduction, and so on. The following is the significance of the proposed research.

6.2.1. Model All Constraints. The proposed approach covers the modeling of various types of constraints, such as simple OCL constraints, and functional and security crosscutting constraints.

6.2.2. Reduce Maintenance. The maintenance of design models and constraint aspects is reduced. The constraints are separately managed and modified without affecting the design models. For example, if we want to modify the balance addition and deduction constraints, we can easily modify them without altering the design models. In this way, the rate of manageability and maintenance is also reduced.

6.2.3. Improved Separation of Concerns. Our approach has also improved the separation of authentication concerns and their constraints. These concerns are modularized separately from functional logic.

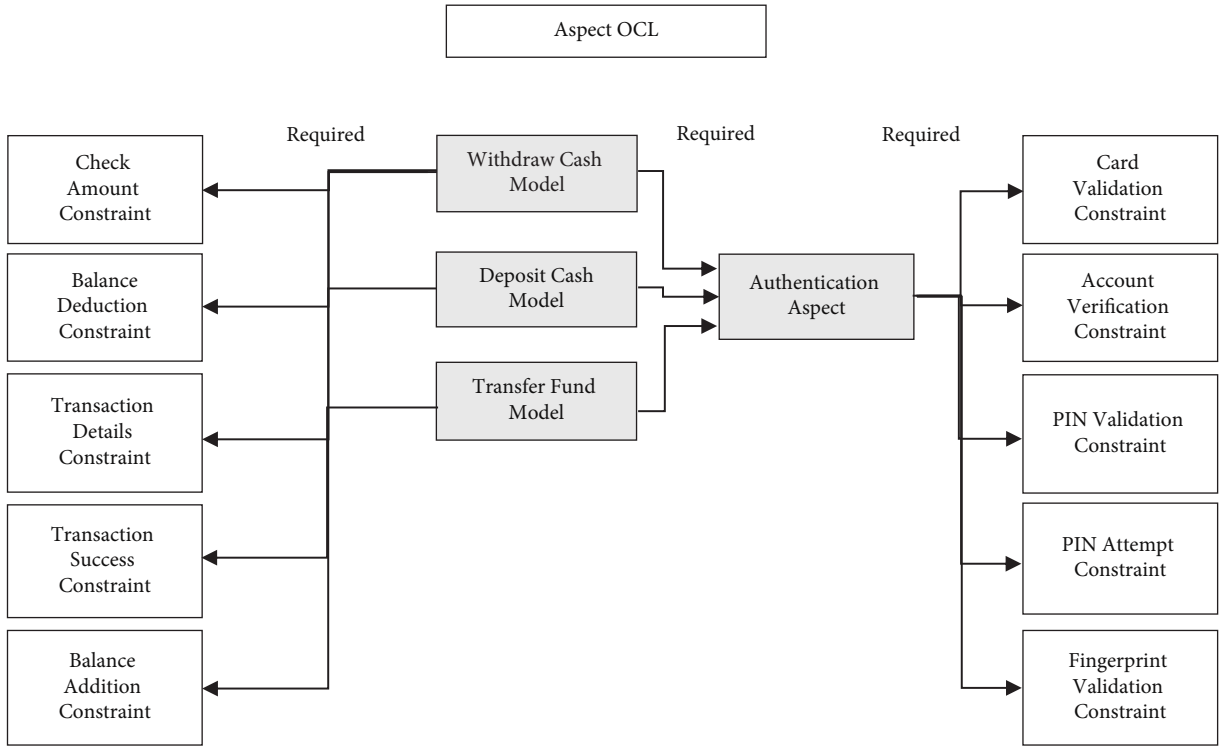

Figure 24: Constraints specification using aspect OCL.

Table 3: Limitations of the previous approaches.

Papers	Functional requirements modeling	Crosscutting quality attributes modeling (security, reliability, etc.)	Separation of simple and crosscutting constraints	Extension of aspect OCL constraint	Crosscutting constraints modeling (both functional and authentication)	Weaving of crosscutting constraints	Verification of constraint-oriented models
Shaukat et al. [1]	✓	✓					
Djedjiga et al. [6]	✓	✓					
Indrakshi et al. [9]	✓	✓					
Ubaid et al. [10]	✓	✓					
Kendra et al. [22]	✓	✓					
Geri et al. [23]	✓	✓					
Nada et al. [24]	✓	✓					
Dianxiang et al. [25]	✓	✓					
Xiang and Li [26]	✓	✓					
Guisheng et al. [27]	✓	✓					
Lidia et al. [28]	✓	✓					
Our approach	✓	✓	✓	✓	✓	✓	✓

6.2.4. Reduction in Complexity. The reduction in the number of duplicate constraints reduced the time and effort required for maintenance, which, accordingly, also reduced the complexity of constraints and design models.

6.2.5. Better Readability. The proposed method improved the readability and understandability of the design models and constraints. The modularized models and constraints are easily comprehensible.

6.2.6. Generalization. The results can also be generalized to other security aspects, i.e., authorization, integrity, confidentiality, etc.

7. Conclusion

Modeling the security mechanism to prevent security vulnerabilities and threats is highly essential during software design. In recent years, various researchers have used the aspect-oriented technique for security modeling. These researchers have only solved the problems of crosscutting security behavior in various design artifacts such as state machines, activity diagrams, etc. but ignored the problems of security crosscutting constraints, specifically authentication crosscutting constraints. These constraints are difficult to maintain and manage and result in duplication when applied to design artifacts using OCL. To cope with these problems, we presented an authentication-oriented modeling approach to correctly address the crosscutting constraints as independent units using aspect OCL. The proposed approach successfully removed the constraints' tangling and scattering from the sequence diagram during the design stage. The constraint-oriented resultant models are successfully verified. Moreover, the results show that the constraints are reduced using aspect OCL, which consequently reduces the maintenance, management, and complexity of these constraints. In the future, we intend to develop a tool for the automated weaving of constraints into design models.

Data Availability

The data collected during the data collection phase are available from all the authors upon request. The data can be obtained from publicly available repositories using the DOIs mentioned in the paper.

Conflicts of Interest

No potential conflicts of interest have been declared.

Authors' Contributions

Writing–original draft: Ubaid Ullah, Writing–review & editing: Ubaid Ullah, Conceptualization: Ubaid Ullah, Investigation: Ubaid Ullah, Methodology: Ubaid Ullah, Validation: Ubaid Ullah, Data acquisition: Ubaid Ullah, Software & resources: Ubaid Ullah, Additional review: Usama Musharaf, Funding acquisition: Muhammad Haleem. Correspondence: Muhammad Haleem.

Acknowledgments

This study is supported and funded by the corresponding author's institution, Kardan University, Afghanistan. No additional external support was received for this study.

References

[1] S. Ali, L. C. Briand, and H. Hemmati, "Modeling robustness behavior using aspect-oriented modeling to support robustness testing of industrial systems," *Software and Systems Modeling*, vol. 11, no. 4, pp. 633–670, 2012.

[2] M. Iqbal, "Nfr modeling approaches," in *Proceedings of the 2011 First ACIS International Symposium on Software and Network Engineering*, pp. 109–114, IEEE, 2011.

[3] P. Salini and S. Kanmani, "Security requirements engineering process for web applications," *Procedia Engineering*, vol. 38, pp. 2799–2807, 2012.

[4] D. Mouheb, M. Debbabi, M. Pourzandi et al., *Aspect-oriented security hardening of UML design models*, Springer International Publishing, 2015.

[5] D. Mouheb, C. Talhi, M. Nouh et al., "Aspect-oriented modeling for representing and integrating security concerns in UML," in *Software Engineering Research, Management and Applications 2010*, pp. 197–213, Springer, Berlin, Heidelberg, 2010.

[6] D. Mouheb, D. Alhadidi, M. Nouh, M. Debbabi, L. Wang, and M. Pourzandi, "Aspect-oriented modeling framework for security hardening," *Innovations in Systems and Software Engineering*, vol. 12, no. 1, pp. 41–67, 2016.

[7] M. Dutta, Kangkhita Keam Psyche, and S. Yasmin, "ATM transaction security using fingerprint recognition," *Am J Eng Res (AJER)*, vol. 6, no. 8, pp. 2320–0847, 2017.

[8] M. Khan, U. Ahmed, and M. Zulkernine, "On selecting appropriate development processes and requirements engineering methods for secure software," in *33rd Annual IEEE International Computer Software and Applications Conference*, vol. 2, pp. 353–358, IEEE, 2009.

[9] I. Ray, R. France, Na Li, and G. Georg, "An aspect-based approach to modeling access control concerns," *Information and Software Technology*, vol. 46, no. 9, pp. 575–587, 2004.

[10] U. Ullah, R. B. Faiz, and M. Haleem, "Modeling and verification of authentication threats mitigation in aspect-oriented mal sequence woven model," *PLoS One*, vol. 17, no. 7, Article ID e0270702, 2022.

[11] M. Völter, T. Stahl, J. Bettin, A. Haase, and H. Simon, *Model-driven Software Development: Technology, Engineering, Management*, John Wiley & Sons, 2013.

[12] M. Brambilla, J. Cabot, and M. Wimmer, "Model-driven software engineering in practice: second edition," *Synthesis lectures on software engineering*, vol. 3, no. 1, pp. 1–207, 2017.

[13] J. Warmer and A. Kleppe, *The Object Constraint Language: Precise Modeling with*, UML Addison-Wesley, Reading, MA, 1998.

[14] T. Stahl, M. Völter, and K. Czarnecki, *Model-driven Software Development: Technology, Engineering, Management*, John Wiley & Sons, 2006.

[15] M. U. Khan, H. Sartaj, M. Z. Iqbal, M. Usman, and N. Arshad, "Aspectocl: using aspects to ease maintenance of evolving constraint specification," *Empirical Software Engineering*, vol. 24, no. 4, pp. 2674–2724, 2019.

[16] G. C. Murphy, R. J. Walker, E. L. A. Baniassad, M. P. Robillard, A. Lai, and M. A. Kersten, "Does aspect-

oriented programming work?" *Communications of the ACM*, vol. 44, no. 10, pp. 75–77, 2001.

[17] S. Ali, T. Yue, and L. C. Briand, "Does aspect-oriented modeling help improve the readability of UML state machines?" *Software and Systems Modeling*, vol. 13, no. 3, pp. 1189–1221, 2014.

[18] M. U. Khan, N. Arshad, M. Z. Iqbal, and H. Umar, "AspectOCL: extending OCL for crosscutting constraints," in *European Conference on Modelling Foundations and Applications*, pp. 92–107, Springer, Cham, 2015.

[19] C. study Cigital, *Finding Defects Early Yields Enormous Savings*, White paper), 2003.

[20] K. Hoo, "Tangible ROI through secure software engineering," *Security Business Quarterly*, 2001.

[21] I. Tarandach and J. Matthew, *Coles. Threat Modeling*, O'Reilly Media, Inc, 2020.

[22] K. Cooper, L. Dai, and Yi Deng, "Performance modeling and analysis of software architectures: an aspect-oriented UML based approach," *Science of Computer Programming*, vol. 57, no. 1, pp. 89–108, 2005.

[23] G. Georg, I. Ray, K. Anastasakis, B. Bordbar, M. Toahchoodee, and S. H. Houmb, "An aspect-oriented methodology for designing secure applications," *Information and Software Technology*, vol. 51, no. 5, pp. 846–864, 2009.

[24] N. H. Sherief, A. A. Abdel-Hamid, and K. M. Mahar, "Threat-driven modeling framework for secure software using aspect-oriented Stochastic Petri nets," in *Proceedings of the 2010 the 7th International Conference on Informatics and Systems (INFOS)*, pp. 1–8, IEEE, Cairo, Egypt, March 2010.

[25] D. Xu and K. E. Nygard, "Threat-driven modeling and verification of secure software using aspect-oriented Petri nets," *IEEE Transactions on Software Engineering*, vol. 32, no. 4, pp. 265–278, 2006.

[26] X. Qiu and Li Zhang, "Specifying redundancy tactics as crosscutting concerns using aspect-oriented modeling," *Frontiers of Computer Science*, vol. 8, no. 6, pp. 977–995, 2014.

[27] G. Fan, H. Yu, and L. Chen, "A formal aspect-oriented method for modeling and analyzing adaptive resource scheduling in cloud computing," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 281–294, 2016.

[28] L. Fuentes, N. Gámez, and P. Sánchez, "Aspect-oriented design and implementation of context-aware pervasive applications," *Innovations in Systems and Software Engineering*, vol. 5, no. 1, pp. 79–93, 2009.

[29] F. Rademacher, S. Sachweh, and Z. . Albert, "Aspect-oriented modeling of technology heterogeneity in microservice architecture," in *Proceedings of the 2019 IEEE International Conference on Software Architecture (ICSA)*, pp. 21–30, IEEE, Hamburg, Germany, March 2019.

[30] H. A. Alhamad and M. M. Hassan, "Aspect-oriented models-based framework to secure intelligent systems," in *Proceedings of the 2022 8th International Conference on Computer Technology Applications*, pp. 249–262, 2022.

[31] S. Yang and Z. Wei-Dong, "Aspect-oriented modeling in concurrent system," in *Proceedings of the 2019 IEEE 3rd Information Technology, Networking, Electronic And Automation Control Conference (ITNEC)*, pp. 836–840, IEEE, Chengdu, China, March 2019.

[32] J. Flotyński, "Visual aspect-oriented modeling of explorable extended reality environments," *Virtual Reality*, vol. 26, no. 3, pp. 939–961, 2022.

[33] G.-J. Ahn and M. E. Shin, "Role-based authorization constraints specification using object constraint language," in *Proceedings of the Tenth IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 157–162, IEEE, Cambridge, MA, USA, June 2001.

[34] G.-J. Ahn and R. Sandhu, "Role-based authorization constraints specification," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 207–226, 2000.

[35] R. Sandhu, "Rationale for the RBAC96 family of access control models," in *Proceedings of the First ACM Workshop on Role-Based Access Control*, p. 9, Gaithersburg, Maryland, USA, December 1996.

[36] H. Wang, J. Cao, and Y. Zhang, "Role-based access control constraints and object Constraint Language," in *Access Control Management in Cloud Environments*, pp. 141–158, Springer, Cham, 2020.

[37] R. Sandhu, V. Bhamidipati, and Q. Munawer, "The ARBAC97 model for role-based administration of roles," *ACM Transactions on Information and System Security*, vol. 2, no. 1, pp. 105–135, 1999.

[38] I. S. Bajwa, B. Bordbar, G. Mark, and Lee, "OCL constraints generation from natural language specification," in *Proceedings of the 2010 14th IEEE International Enterprise Distributed Object Computing Conference*, pp. 204–213, IEEE, Vitoria, Brazil, October 2010.

[39] S. Ali, M. Zohaib Iqbal, A. Arcuri, and L. C. Briand, "Generating test data from OCL constraints with search techniques," *IEEE Transactions on Software Engineering*, vol. 39, no. 10, pp. 1376–1402, 2013.

[40] S. Ali, T. Yue, M. Z. Iqbal, and R. Kaur Panesar-Walawege, "Insights on the use of OCL in diverse industrial applications," in *International Conference on System Analysis and Modeling*, pp. 223–238, Springer, Cham, 2014.

[41] O. MG. Ocl, *Object Management Group:'Object Constraint Language (OCL)'*, OMG Available Specification, 2006.

[42] A. D. Brucker, J. Cabot, G. Daniel et al., "Recent developments in OCL and textual modelling," in *International Workshop on OCL and Textual Modeling*, pp. 157–165, OCL, 2016.

[43] D.-H. Dang and M. Gogolla, "An OCL-based framework for model transformations," *VNU Journal of Science: Computer Science and Communication Engineering*, vol. 32, p. 1, 2016.

[44] A. A. Jilani, Z. Muhammad, M. Iqbal, U. Khan, and M. Usman, "Advances in applications of object constraint language for software engineering," in *Advances in Computers*, vol. 112, pp. 135–184, Elsevier, 2019.

[45] U. Ullah, *ATM Case Study Requirements"*, 2022.

[46] U. Ullah E-Commerce CaseStudy", 2022.

[47] E. N. Kasanda and P. Jackson, "ATM security: a case study of emerging threats," *International Journal of Advanced Studies in Computers, Science and Engineering*, vol. 63, 2019.

[48] https://en.wikipedia.org/wiki/Automated_teller_machine#Fraud.

[49] J. Braeuer, B. Gmeiner, and J. Sametinger, "ATM security: a case study of a logical risk assessment," in *ICSEA 2015: The Tenth International Conference on Software Engineering Advances*, 2015.

[50] O. Nathaniel and M. Osuo-Genseleke, "A comparative study of PIN based and three-factor based authentication technique for improved ATM security," *International Research Journal of Engineering and Technology*, vol. 5, no. 5, pp. 3749–3754, 2018.

[51] J. Shubhra, "ATM frauds: detection & prevention," *International Journal of Advances in Electronics and Computer Science*, vol. 4, p. 10, 2017.

[52] N. A. Lal, P. Salendra, and F. Mohammed, *A review of authentication methods*, pp. 246–249, 2016.

[53] P. S. Aithal, *A Study on Multifactor Authentication Model Using Fingerprint Hash Code*, Password and OTP, 2018.