Exemplifying the Aspect-Oriented ADOM Approach: The Check-In Check-Out (CICO) Domain

Iris Reinhartz-Berger and Alexander Gold

Department of Management Information Systems,
University of Haifa, Haifa 31905, Israel
iris@mis.haifa.ac.il, alexgold@013.net.il

Abstract. The purpose of this document is exemplifying the aspect-oriented Application-based DOmain Modeling (ADOM) approach, which aims at enabling the design and representation of aspect families and their weaving rules to application families during the entire development lifecycle. In particular, three types of models, aspect, base, and woven models, are defined and exemplified in different abstraction levels. The domain used in this document is Check-In Check-Out (CICO, which manage operations for item enrollment and signing-out. Examples of applications in this domain include library, video rental, hotel management, and car rental systems.

1 Introduction

The aspect-oriented Application-based DOmain Modeling (ADOM) approach enables defining families of aspects and their weaving rules to families of applications during the analysis and design phases. In other words, the suggested approach supports representation of domain aspects and their weaving rules to applications in certain domain. ADOM [4] is used as a framework for defining aspect, base, and woven models at different abstraction levels, namely application and domain levels.

We distinguish among two types of models: base and aspect models. A *base model* describes an application, a system, or a family of such (i.e., a domain). It can stand alone and exhibit both structure and behavior. An *aspect model* describes the structure and behavior that characterize a particular concern. It is not intended to stand alone but to be woven into base models. An aspect model can be defined in the application or domain layer, respectively specifying a specific concern or a family of concerns. Furthermore, each aspect model can be described as a triple of a concern specification (CS), a match pattern (MP), and a merge guidance (MG). The *concern specification* deals only with issues that are relevant to the concern at hand. The *match pattern* constrains the range of base models to which the aspect is applicable. Finally, the *merge guidance* specifies general guidelines for weaving the given aspect to any applicable base model (according to the match pattern). Note that although the same concern specification may have several pairs of suitable match pattern and merge guidance models, we consider an aspect model as the aforementioned triple. In other words, several aspect models may share the same concern specification with different match patterns (and consequently different merge guidance).

For future usage, we define a third type of models, called *woven model*s, which are achieved after applying the rules specified in the merge guidance of an aspect model on a base model that satisfies the match pattern. Note that the resultant woven models are not required to be manually generated, especially due to their complexity. They are only used as a means for understanding the semantics of the weaving process and the complete system structure and behavior.

Fig. 1 summarizes the main model types in the ADOM-based aspect-oriented approach and the relations between them. Although ADOM can be applied to different modeling languages, all models in this document are specified using UML 2.0 class and sequence diagrams [3].

In order to exemplify the suggested approach, a domain of Check-In Check-Out (CICO) applications [1] is used. These applications manage operations for item enrollment and signing-out. Examples of applications in this domain include library, video rental, hotel management, and car rental systems. The rest of the documents present the models of this example in the context of the aspect-oriented ADOM approach.

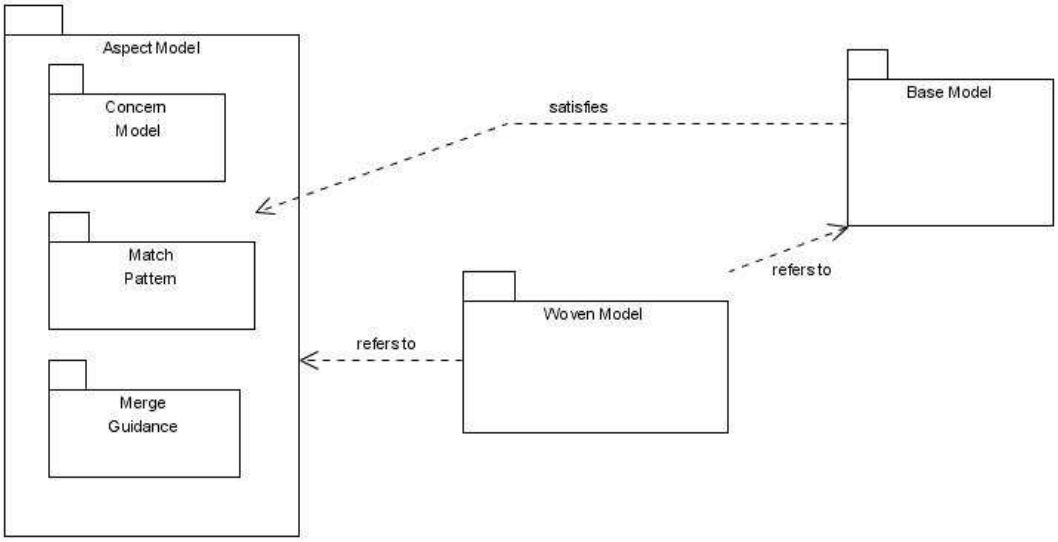

Fig. 1. A package diagram specifying the main model types and relations in the ADOM-based aspect-oriented approach

2 The BASE Models

The base models in the suggested approach are represented using ADOM. The class diagram in Fig. 2 constrains the structure of the CICO domain, namely their main concepts, their required multiplicities, and the relations between them. Any application in this domain, for example, must have exactly one controller with different types of check-in and check-out operations and possible reserve operations; at least one type of items, each of which has one attribute identifying its ID, at least one attribute denoting its status, and at least one descriptive attribute; zero or more item types, as sometimes the reservation in this kind of applications is for item types rather than for individual items; at least one type of loaners, each of which has one attribute identifying its ID and at least one attribute denoting its status, and so on. The abilities to maintain a waiting list in case the item or items are occupied and to arrange the items as collections are optional, as not all the applications in the domain support this functionality.

The sequence diagram in Fig. 3 denotes a typical check-out operation which specifies the procedure of taking or loaning an item (and reserving it if not available). This sequence explicitly refers to whether the items are arranged in collections or not.

Fig. 4 and Fig. 5 exemplify an application model in ADOM-UML which specifies a library system in the CICO domain. This application model is a valid instantiation of the CICO domain model presented in Fig. 2 and Fig. 3. The class diagram in Fig. 4 contains two types of loaners (students and staff members), two types of item types (books and multimedia), one type of items (copies), one type of waiting lists (book reservations), loans, and so on. Note that since the different loaners and the different item types have similarities in the application model, inheritance relations are used. The check-out operation, which is specified in Fig. 5, deals with loaning a book. The objects (lifelines) and procedure calls in this diagram are not stereotyped according to the domain model, as their stereotypes can be deduced from the class diagram.

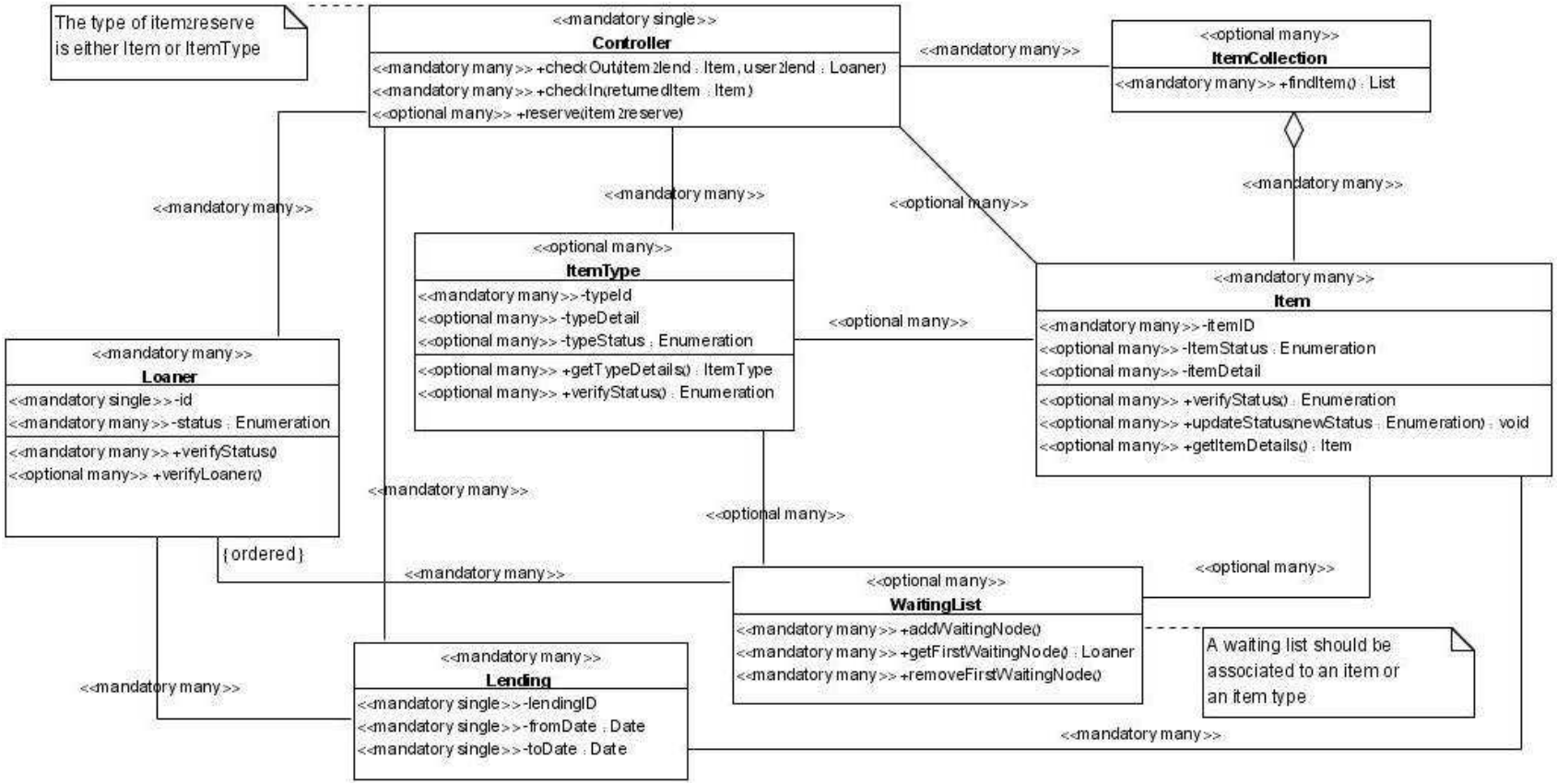

Fig. 2. A class diagram of the CICO domain

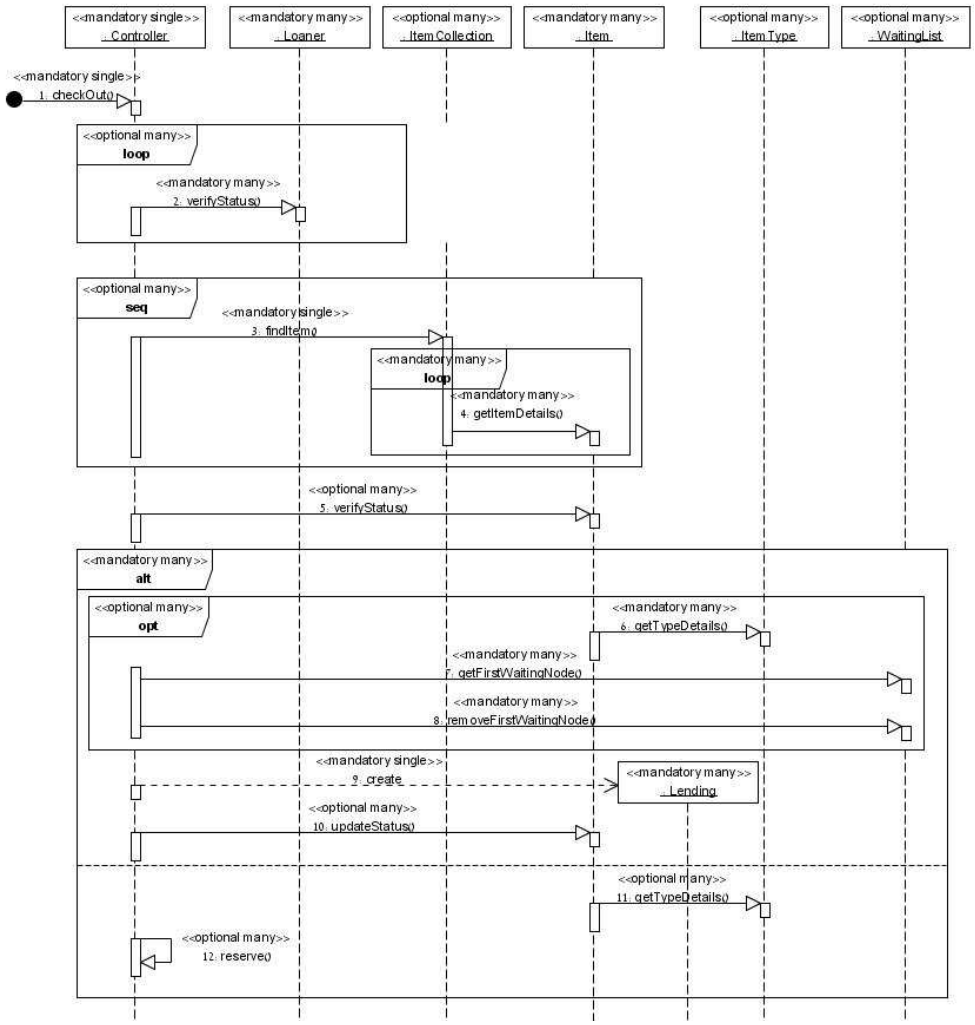

Fig. 3. A (domain) sequence diagram specifying a check-out operation

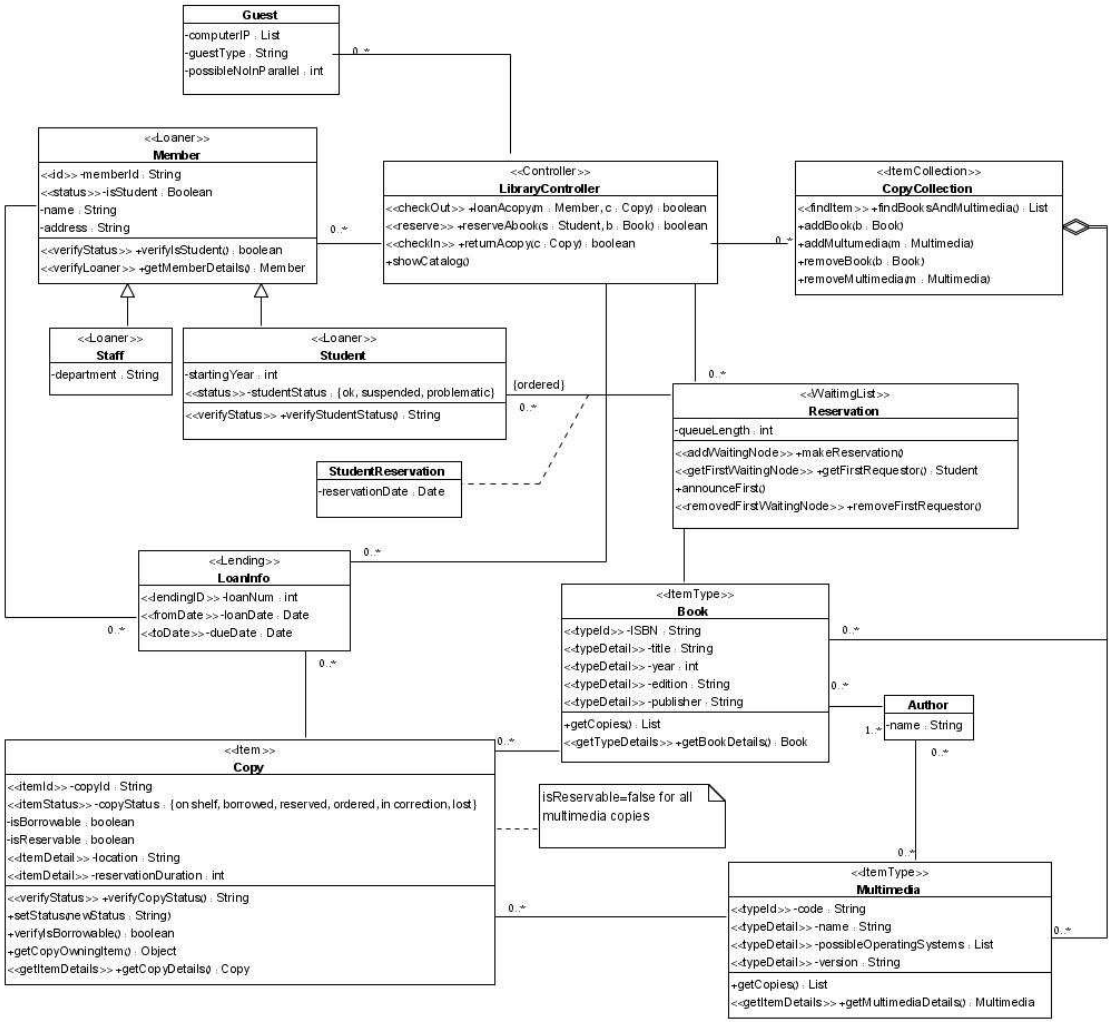

Fig. 4. An application class diagram of the library system

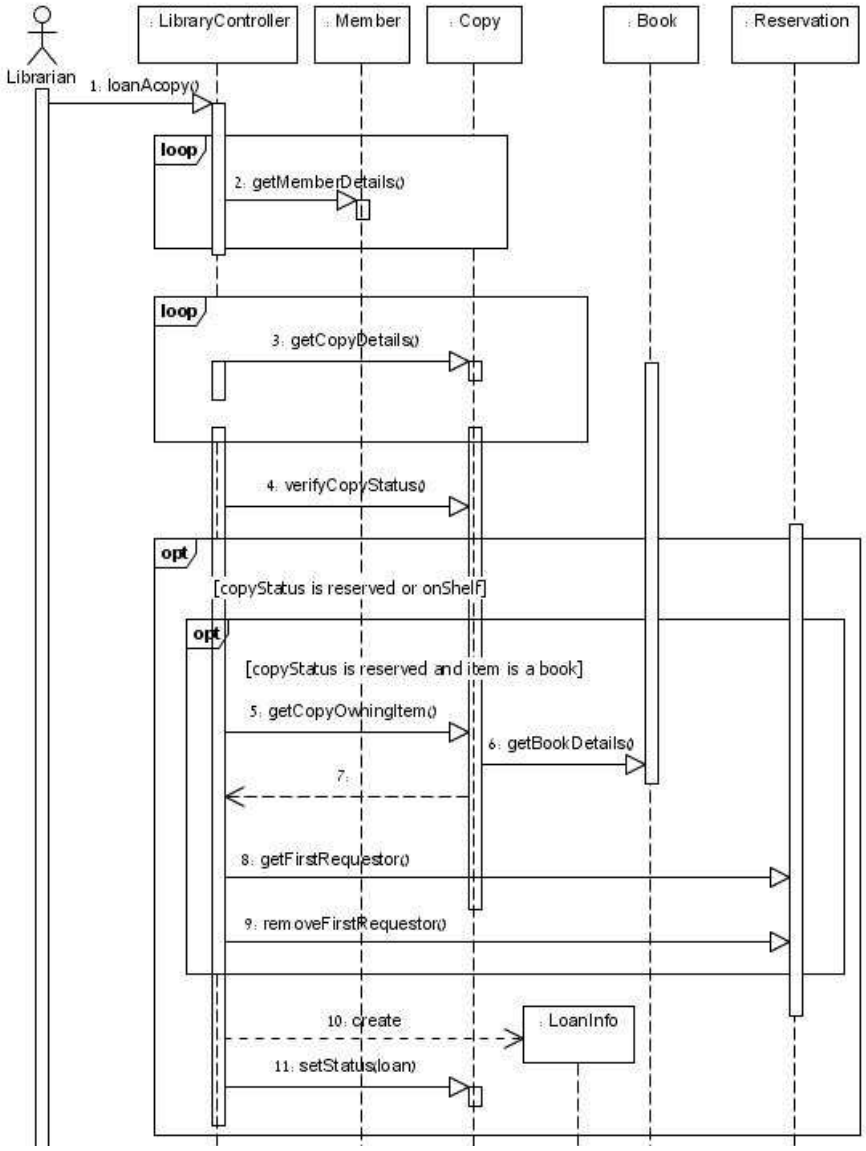

Fig. 5. An application sequence diagram specifying a loan operation

3 The ASPCET Model

3.1 Concern Specification

As an example of a domain aspect (i.e., a family of aspects) consider security. Computer security is a branch of computer science concerned with risk management trade-offs in the areas of confidentiality, integrity, and availability of electronic information that is processed by or stored on computer systems [5]. Systems which contain fundamental flaws in their security designs cannot be made secure without compromising their usability. However, in many cases security techniques can be woven into (existing) system designs and, hence, considered as aspects. Examples of particular security aspects (which reside in the application layer) are: (1) authorization which deals with protecting computer resources by allowing those resources to be used only by consumers that have been granted authority to use them, (2) authentication which is the act of establishing or confirming something (or someone) as authentic, that is, that claims made by or about the thing are true, and (3) fraud protection which deals with protecting someone (or something) from fraud activities, such as theft, false billing, and bait, by recording the history of system activities and analyzing the collected data.

The domain model depicted in Fig. 6 and Fig. 7 describes the commonality and variability allowed in a family of security aspects. Each aspect in this domain must deal with performers, secured items, and Actions. The class diagram in Fig. 6 captures these concepts, specifying their structure and relationships. It also refers to three additional (optional) classes: Policy that may refer to a specific performer, a specific item, or a specific performer-item pair; History that may record security-related activities; and Analyzer that may be used, for example, for detecting different threats. Using UML sequence diagram notation, Fig. 7 describes how the allowance of a secured activity is checked. This sequence can be used in different contexts, as is demonstrated latter.

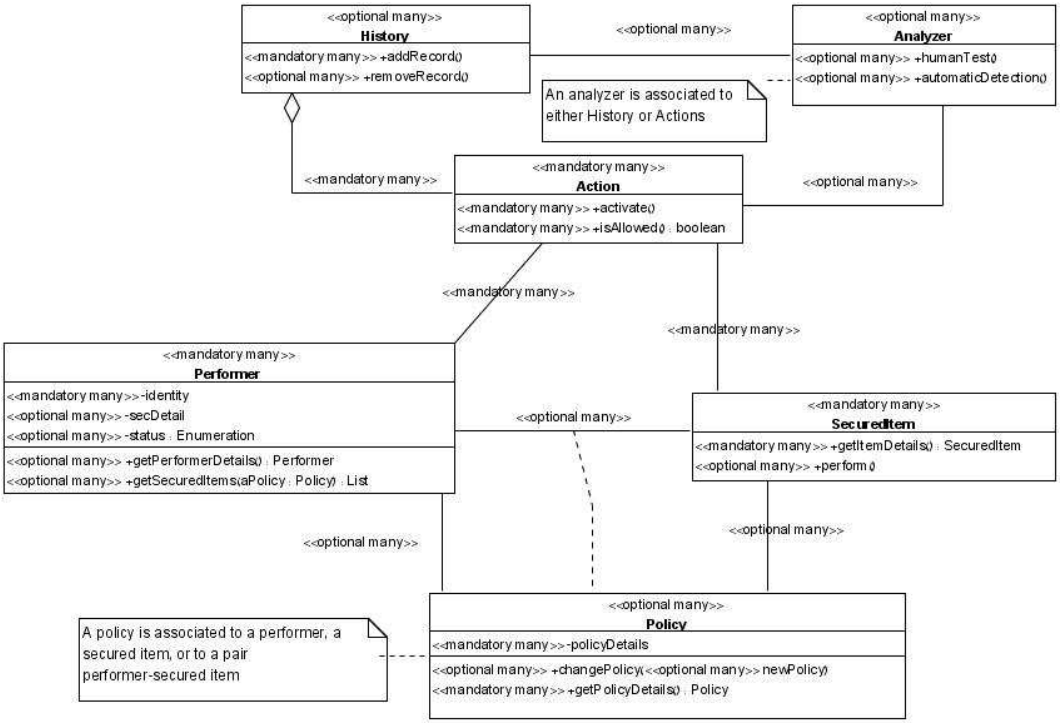

Fig. 6. A domain class diagram of the security aspect family

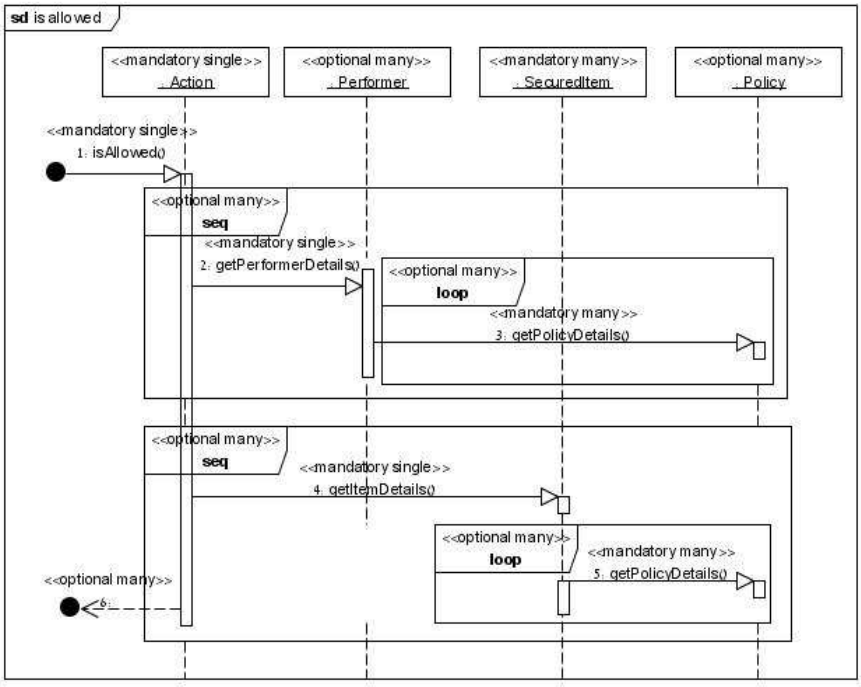

Fig. 7. A domain sequence diagram describing action activation in the security aspect family

Fig. 8 and Fig. 9 exemplify a particular authorization aspect, which enables executing only authorized actions by users. This authorized aspect is an instantiation of the security (domain) aspect. The sequence diagram in Fig. 9(a) checks if the user has the rights to perform a certain action on an item, while Fig. 9(b) embeds this sequence in another sequence which executes the secured action if the user is authorized to do it.

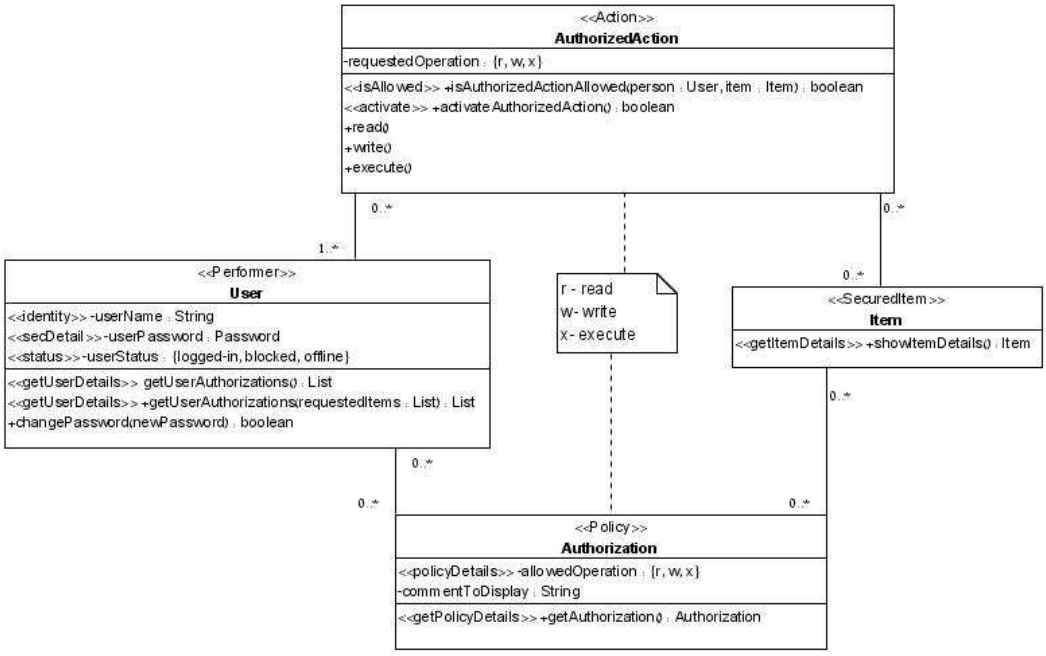

Fig. 8. An application class diagram of an authorization aspect

3.2 Match Patterns

A match pattern is the part of an aspect model that specifies structural and behavioral rules and constraints on the base models to which the concern specification can be woven. In other words, this part represents the minimal requirements from the base models that can weave the concern specification into them. Furthermore, as explained in the next section, a match pattern defines "anchors" (join points) to which the merge guidance can refer. The least restricting match pattern is the empty model which implies that the aspect model in general and its concern specification in particular are applicable and can be woven to any base model. Making the match pattern more detailed reduces the number of base models to which the concern specification can be woven, but enables specifying more reasonable and detailed weaving rules.

For visually specifying match rules in ADOM-UML, we define in the language layer a <<matchcond>> stereotype, which is associated to the top level *Element* metaclass in the UML metamodel and has two associated tagged values: elCardinality and elConstraint. elCardinality specifies the range of elements required to be matched to this element in the base model, whereas elConstraint constrains the possible matched elements using OCL [2]. The default values of these tagged values (when not presented) are the less restricting ones, namely elConstraint is true and elCardinality is optional many (0..n).

As an example, Fig. 10 exemplifies a match pattern of the security (domain) aspect for domain base models. This match pattern requires that the base model to which the concern specification will be woven includes Items with operations that return void and are specified both structurally (in the class diagram) and behaviorally (at least in one sequence diagram). Furthermore, the base model may have a Controlling Element, which is connected to an Item via an association and activates its operationOnItem method.

The CICO domain model specified in Fig. 2 and Fig. 3 satisfies the match pattern in Fig. 10. In particular, Controller from CICO corresponds to Controlling Element and Item from CICO corresponds to Item of the match pattern, while updateStatus corresponds to operationOnItem. The other two operations of Item do not match operationOnItem, since they do not return void. ItemType and ItemCollection do not match Item (from the match pattern) since they do not have "void" operations. Finally, the association Controller-Item corresponds to controlledItems.

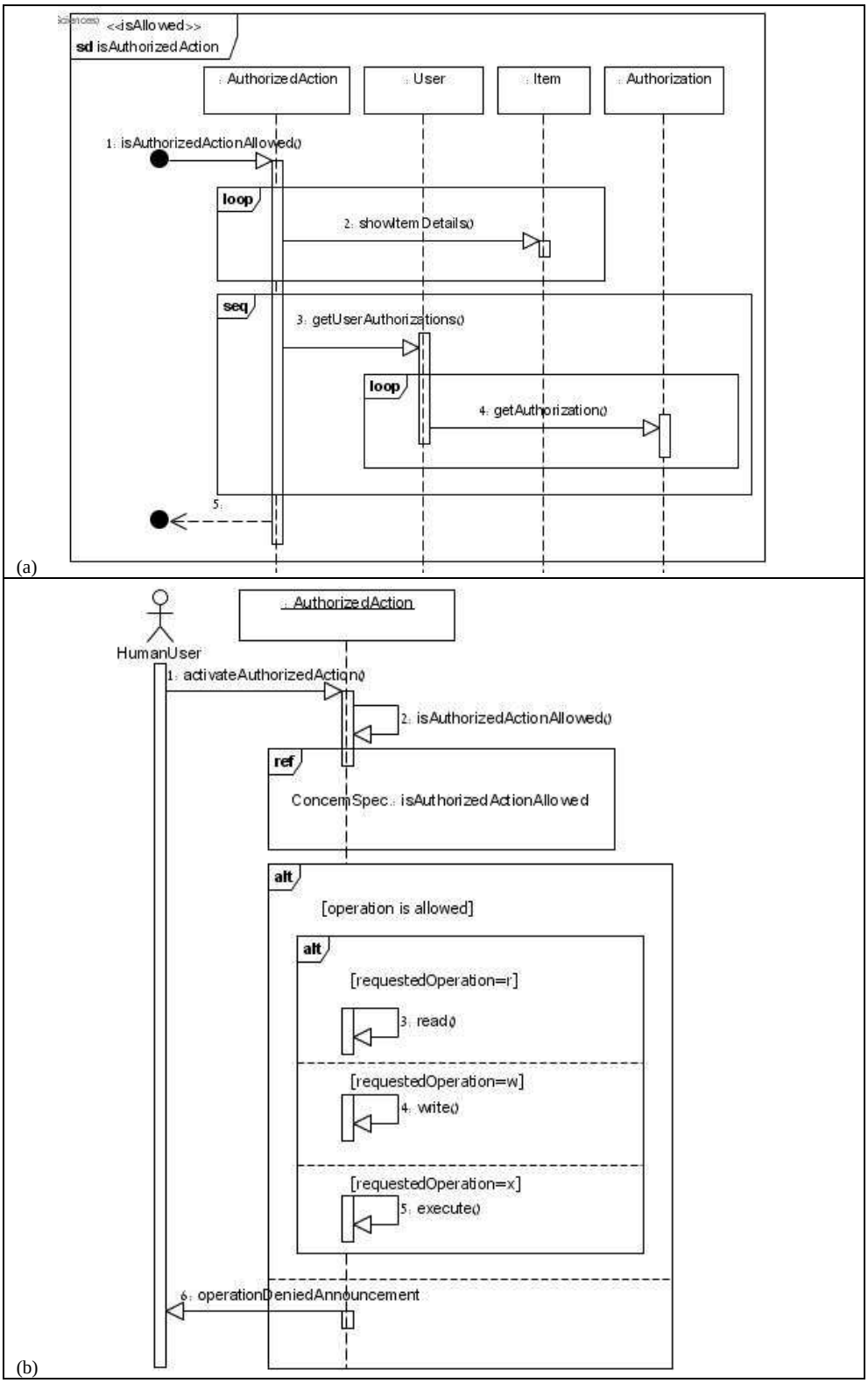

Fig. 9. (a) An application sequence diagram of checking the possibility to activate an authorized action. (b) An application sequence diagram of activating an authorized action.

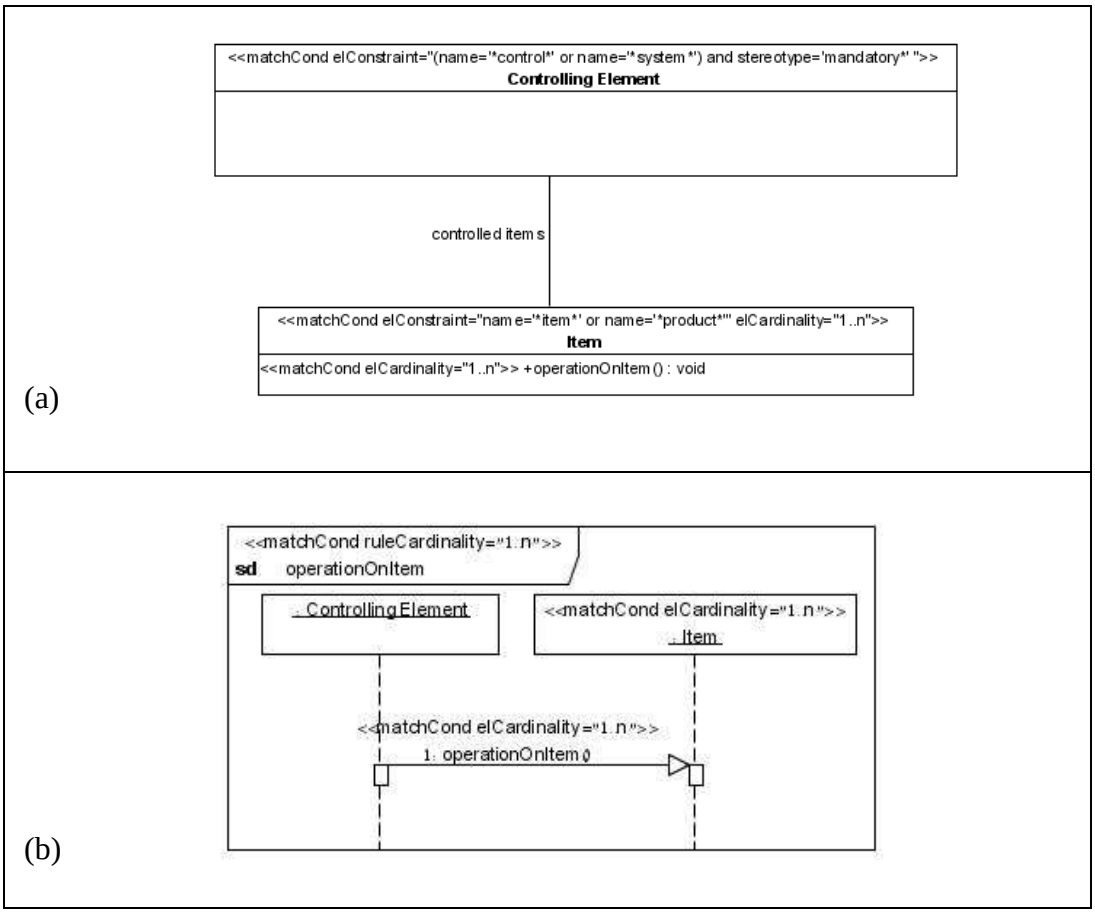

Fig. 10. The match pattern for the security aspect and domain base models: (a) the structural constraints specified as a class diagram and (b) the behavioral constraints specified as a sequence diagram.

3.3 Merge Guidance

The merge guidance of an aspect model combines the concern specification and the match pattern of the same aspect in order to guide the designer how to weave the concern specification into a base model that fulfills the match pattern rules. For this purpose, the elements of the match pattern and the concern specification are used as the elements of the merge guidance[1].

We distinguish between four types of operations: combining, concern addition, merge addition, and match only operations. A *combining operation* takes two elements, one from the concern specification (CS) and the other from a match rule in the match pattern (MP), and combines them into a third element that exhibits the features of the two composing elements. A *concern addition operation* enables the concern specification adding elements that does not exist neither have counterparts in the base model. A *merge addition operation* enables adding elements that do not appear in the concern specification neither in the match pattern, but are rather required when merging or weaving the concern specification into a base model that satisfies the match rule. Finally, a *match only operation* enables specifying elements that are required only for matching base models, but are not modified as a result of weaving the concern specification into the base model.

As an example consider the concern specification depicted in Fig. 6 and Fig. 7 and the match pattern specified in Fig. 10. A possible merge guidance *MG* is depicted in Fig. 11. The different e_i are name replacers. Fig. 11(a) combines SecuredItem from the concern specification with Item from the match pattern (where perform is combined with operationOnItem). The merge guidance further adds an association between Controlling Element (if it exists in the base model) and Action, History, or Analyzer (from the concern specification). Fig. 11(b) adds to the above guidance the information that the sequence titled "is allowed", which is depicted in Fig. 7, should

[1] We assume that the name spaces of the concern specification and the match pattern of the same aspect are distinctive, otherwise adding the model (package) name to the element names is required.

be embedded exactly before any "void" operation on an item (as the sequence combined fragment indicates). This merge guidance can be used in order to weave any security aspect (e.g., authorization) to any application in the CICO domain (e.g., the library system).

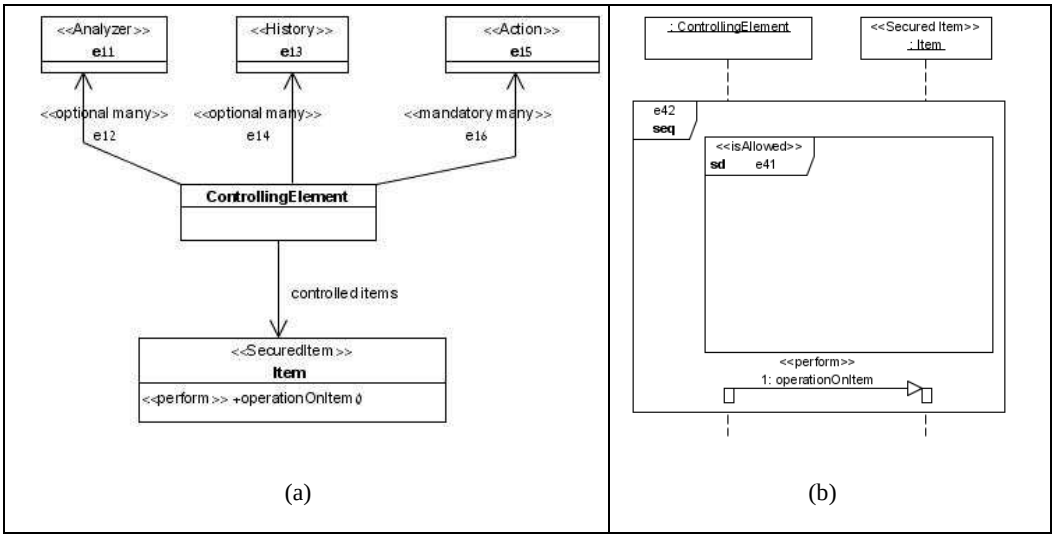

Fig. 11. The merge guidance of the security aspect to domain base models: (a) the structural merge specified as a class diagram and (b) the behavioral merge specified as a sequence diagram.

4 The WOVEN Model

The woven model is achieved by finding matches between a base model and a match pattern and replacing each such occurrence with the merge guidance. Only maximal matches are used for this purpose, i.e., matches that any model parts that include them are not considered as matches. Note that there may be more than one maximal match in a given base model that satisfies a single merge rule, implying application of the same merge rule several times to the base model (with different model portions).

Fig. 12 to Fig. 13 present the woven model resulted after weaving the security (domain) aspect into the CICO (domain) base model. In these figures, the elements that belong only to the base model are depicted in white, the base model elements that are combined with aspect elements are depicted in bold and grey, and the elements that are added due to the aspect or merge guidance are depicted in grey. As mentioned, the woven model was generated only for the purpose of comprehending better the meanings of the aspect model parts. In the resultant woven model, the terminology (i.e., element names) is first taken from the base model and only afterwards (for additions) from the aspect model (or more accurately, the merge guidance).

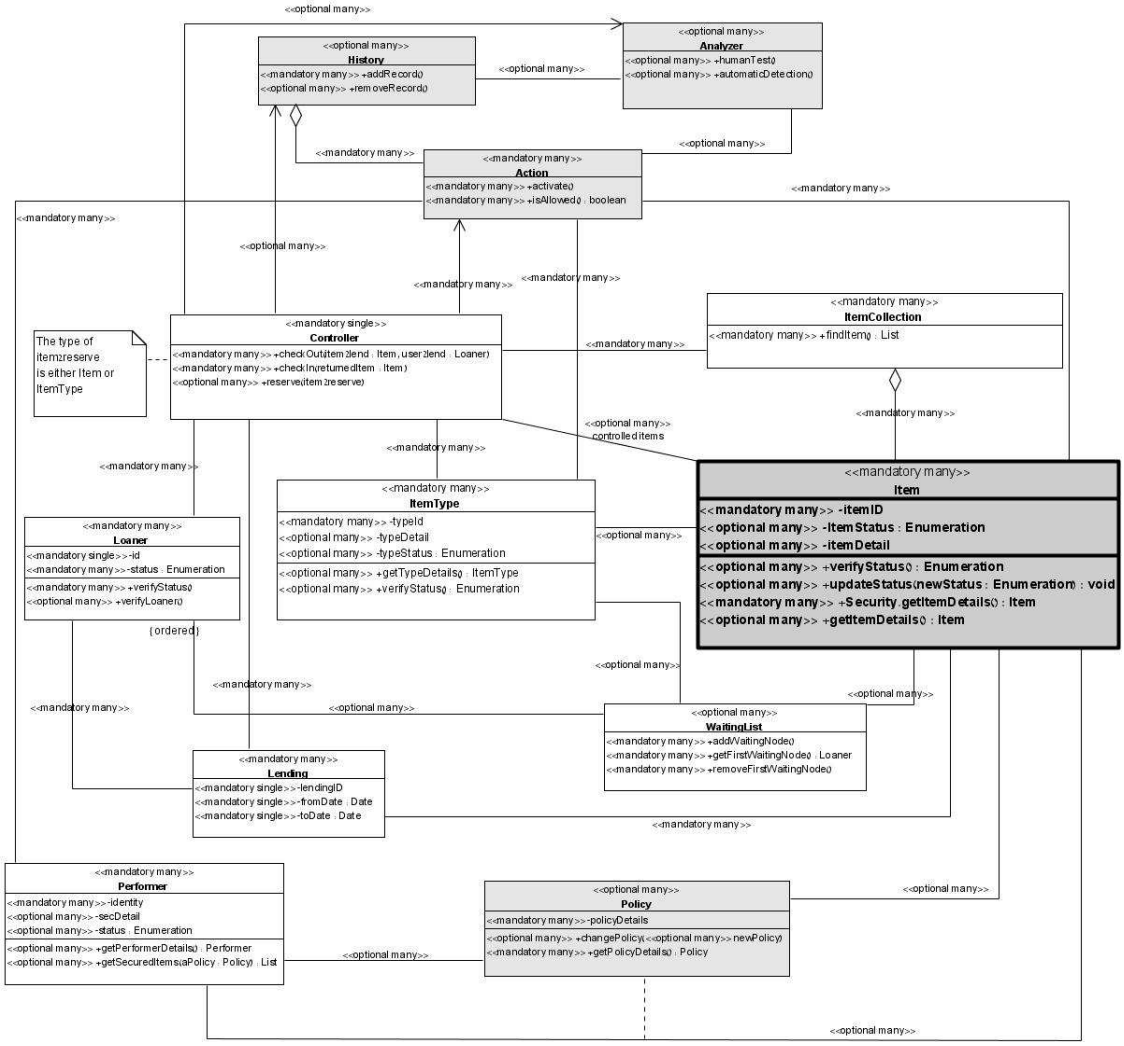

Fig. 12. The woven model resulted after weaving the security aspect into the CICO model. Part A: the structural specification modeled as a class diagram[2]

2 Note that Performer was not combined with Loaner in this model and the two getItemDetails methods of Item were not combined, since these kinds of merging were not explicitly specified in the merge guidance.

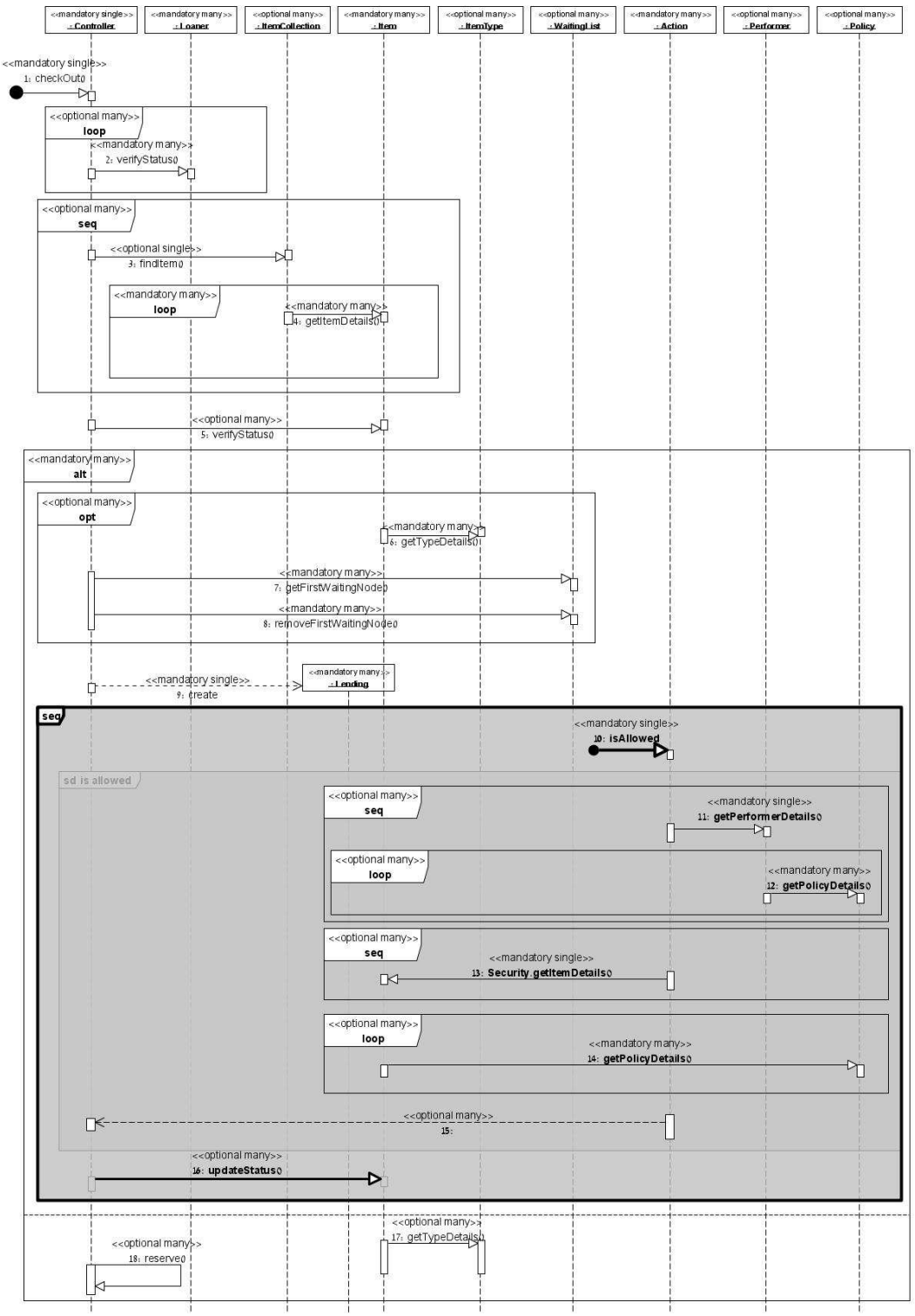

Fig. 13. The woven model resulted after weaving the security aspect into the CICO model. Check-out behavior specified as a sequence diagram

4 Percolating Domain Merge Guidance to Application Models

Having domain level aspect and base models, one can use them for weaving particular aspects into specific applications. In particular the domain match pattern and merge guidance models can assist in application weaving processes. Fig. 14 to Fig. 15, for example, depict the domain-derived application woven model of the library application with respect to the authorization aspect.

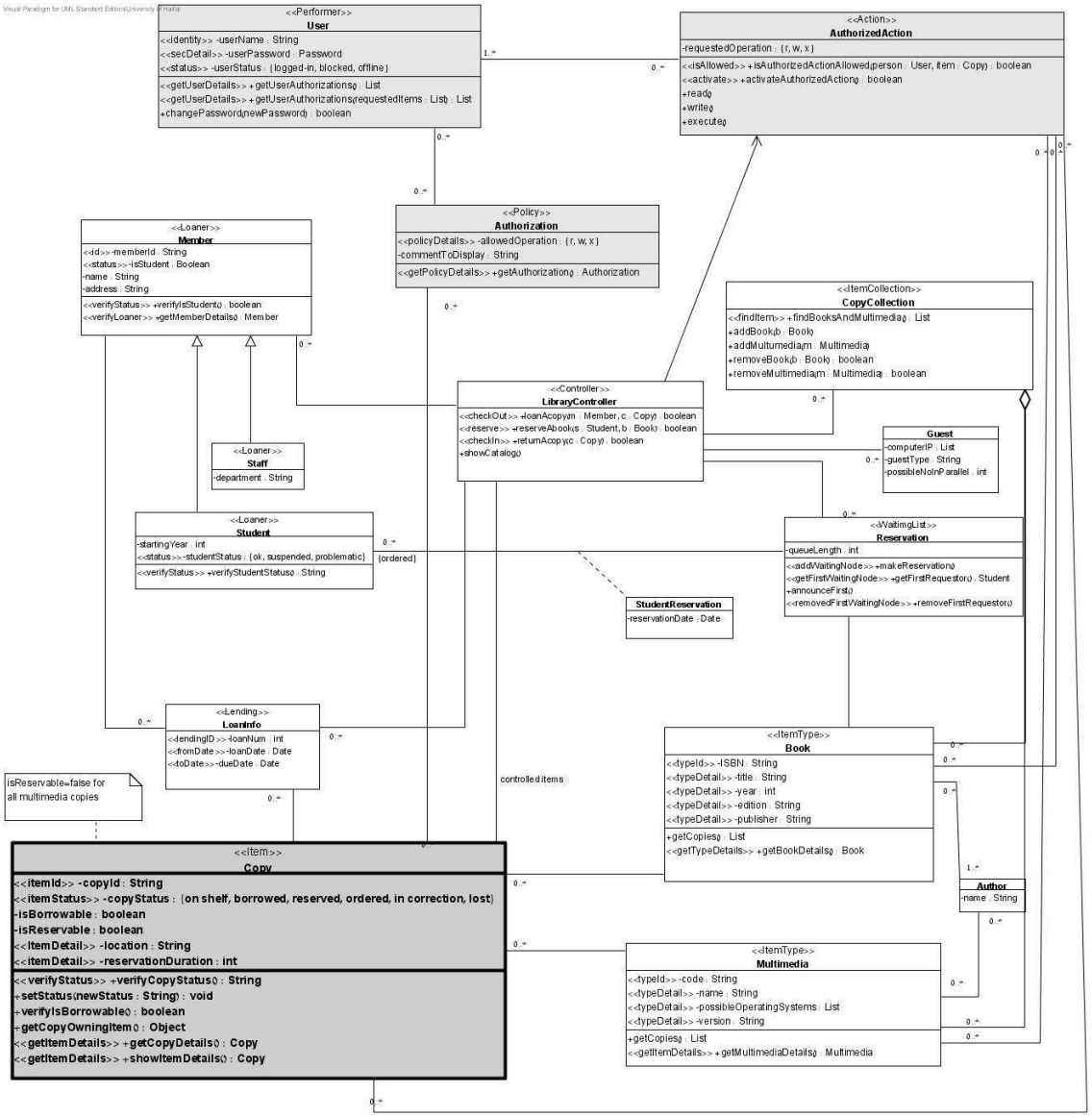

Fig. 14. The domain-derived application woven model of the Library system with respect to the authorization aspect. The structural specification modeled as a class diagram.

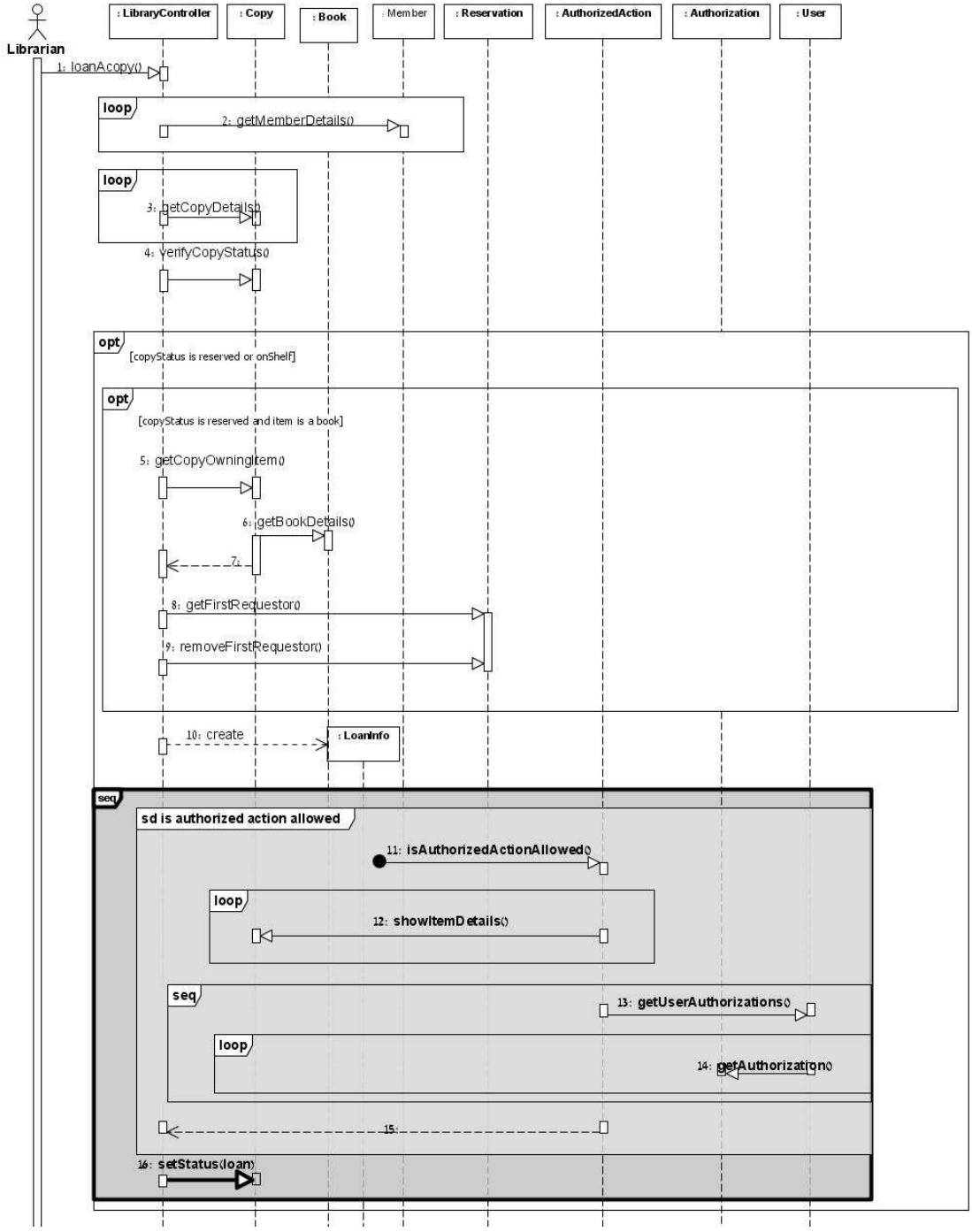

Fig. 15. The domain-derived application woven model of the Library system with respect to the authorization aspect. The behavior specified as a sequence diagram

References

1. Kim, D., France, R., and Ghosh, S. A UML-Based Language for Specifying Domain-Specific Patterns. Journal of Visual Languages and Computing, Special Issue on Domain Modeling with Visual Languages 15(3-4), 2004, pp. 265-289.
2. OMG. UML 2.0 OCL Specification, 2004, http://www.omg.org/docs/ptc/03-10-14.pdf
3. OMG. Unified Modeling Language: Superstructure, Version 2.0, 2005, http://www.omg.org/docs/formal/05-07-04.pdf
4. Reinhartz-Berger, I. and Sturm, A. Enhancing UML Models: A Domain Analysis Approach. Journal on Database Management (JDM), 19(1), special issue on UML Topics, 2008, pp. 74-94.
5. Wikipedia. Security engineering, 2007, http://en.wikipedia.org/wiki/Security_engineering