Building Security Requirement Patterns for Increased Effectiveness Early in the Development Process

Dan Matheson[1]   Indrakshi Ray[1]   Indrajit Ray[1]   Siv Hilde Houmb[2]
[1]Colorado State University
Computer Science Department
{matheson, iray, indrajit}@cs.colostate.edu
[2]Norwegian University of Science and Technology
Computer Science Department
sivhoumb@idi.nt

Abstract

This paper explores the representation of security concerns and their interactions appropriate for a Model Driven Development approach. The focus is on the representation of the security concerns early in the development process and as abstract forms easily related to the security aspects of the solution requirements, but in a manner that allows for the controlled refinement into a solution. This approach uses UML as a rigorous mechanism to represent the early security concerns and their families of solutions. The security concerns are represented as sets of patterns in UML. Stereotypes and tagged values are used as a mechanism to support requirement traceability during solution development. The traceability mechanisms along with common concepts provide a basis for verifying the adherence of the solution to the requirements. The rigorous nature of UML allows for automatic analysis of imprecise specification earlier in the development process.

1. Introduction

Security features in a system may often interact to give rise to undesirable behavior. Consider, for example, the problems arising from the conflict between confidentiality of data and security-auditing of actions. Confidentiality requires that certain data remain hidden from certain users while security-auditing may require that such data be made available to these users. Such conflict often arises from the different requirements that are produced by different people. Additionally, in the solution development process, the creation of the code for these capabilities is done by different engineers. A good design needs to strike the proper balance between them so that sensitive data does not appear in a log file as a result of poor communication between the development engineers.

In this work we investigate the problem of suitably representing security concerns in the software development process such that their interactions can be properly identified and analyzed. Ideally, this should be accomplished early in the software development process so that the overall complexity and cost of the development is reduced. The approach we propose is to use models to represent the various security concerns and then use a Model Driven Development (MDD) process such as Aspect-Oriented Modeling (AOM) to create the solution. Since models can exist at various levels of abstraction, we can build a set of related security models, which constitute a development pattern for solving a security concern. Expressing the security requirements in a modeling language as early as possible makes the tracking and refinement into a solution easier.

2. Security Concerns

Phrases in requirements like "access is limited to second level managers and up" or from regulatory sources like "employee compensation data must be transmitted to the IRS intact" are sources of security concerns. Several categories of security concerns need to be identified if they are to be handled effectively. The following list covers most commercial software design situations:

- Identification and Authentication

- Authorization and Access Control
- Data Integrity
- Confidentiality or Data Privacy
- Auditing
- Data Authenticity
- Survivability
- Non-repudiation

The categories provide an organization framework for the security concern models and a more precise set of terms for expression of the requirements.
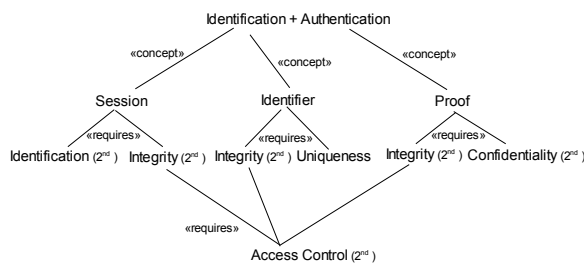
Security Concern Relationships

The security concerns are not independent of each other. By organizing the concerns into categories we are more easily able to represent the relationships between them. There are structural relationships such as *depends* and *used by*, and there are behavioral relationships like *conflicts*. The relationships can be modeled and therefore checked during the model refinement of solution development.

Security Domain Concepts

Starting with the main categories of security concerns we can begin to refine the terms. The refinement results in concepts emerging that were previously hidden by abstraction. For example, session emerges from authentication refinements and auditing refinements. We can create a graphical structure of the related terms as a starting point for discovering patterns.

Figure 1 shows how concepts can emerge. Session emerges as a concept needed to support Identification and Authentication. The same term, Identification, can emerge as a second order concept to support Session.

Figure 1 Emerging Security Concepts

The emergent domain concepts that appear during refinement allow another way to relate the security concerns. With respect to the requirements the common domain concepts should be used in a consistent way across the solution.

The domain concepts involved in a solution will also vary across different engineering realizations. Early in the design process one can use the number and complexity of the domain concepts associated with a particular solution to make cost estimates. Figure 2 shows a first order view of shared solution concepts.

Figure 2 Shared Security Concepts
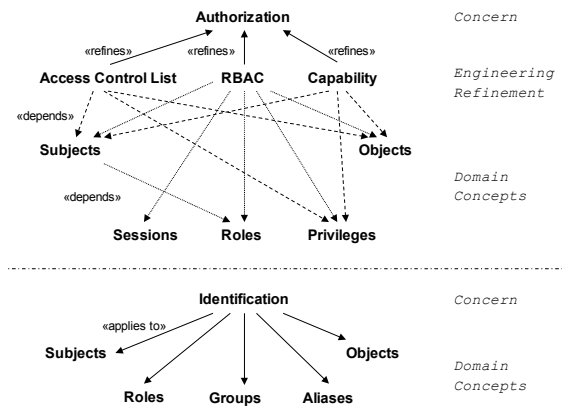
The domain concepts are entities that will need to be realized in a solution. We can model them as classes in UML to represent the concepts and bring them into the solution design. The informal relationships expressed as lines in Figures 1 and 2 can be modeled as associations in the UML.

4. Security Aspect UML Definitions

The representation in UML of the security concerns is first done as class models. As we are focused on the representation of requirements and the early stages of the design, the UML classes have no attributes or methods. The mere presence of an artifact in the class model registers the existence of a requirement.

From an AOM support perspective as well as for capturing domain knowledge we wish to create a set of security domain patterns as UML models at various levels of abstraction. The patterns will exist as class models.

The patterns we are creating are used for several purposes. The first purpose is a starting point for a design to fulfill a specific security concern. The second purpose is as a definition for analysis tools to use when checking the results of a refinement or AOM composition action. A third purpose for an AOM approach is to determine the best order of composition.

The dependency type relationship is used to construct a more complete security solution pattern that matches to a higher level expression of a security concern. For example, our security concern is

confidentiality. We construct an abstract model of the concern that captures the primary concepts of maintaining confidentiality. We know that supporting concepts of user identification and authentication are key to achieving that goal. The more complete confidentiality model will have the dependency relationship to the identity model created.

An AOM composition tool can use the extended solution model as directives for ordering the aspect composition with the primary model. An analysis tool can refer to the confidentiality concern solution model as an extended specification to be tested against.

UML Mechanisms

Several UML mechanisms will be used in the models of security concerns to organize and track the model entities through the MDD refinement process.

One stereotype on classes will be used to track the category of the security concern. This provides a basis for identifying commonalities as models are refined. It can also help in the development of consistency checks. The stereotype will follow the refinement so that the more concrete entities retain the stereotype of the abstract parent. The stereotype embodies a category constraint on the refinement.

A stereotype for domain concepts will also be used. This stereotype will follow refinements in the same manner as the concern category stereotype listed earlier. The purpose of this stereotype is to trace the realization of a security requirement concern to the detailed classes that implement this aspect of the solution.

The AOM approach composes several models together to create a solution at some level of abstraction. The models composed should be at approximately the same level of abstraction or the result is likely to be wrong. We use a tagged value to track the refinement level of a model. If the levels differ by too much, then we are combining models with different levels of precision which can produce a nonsense result.

The composition of models also results in the combining of stereotypes in a result class. This supports the traceability back to the original requirements. It can also be used by analysis tools to check for consistency, conflicts and completeness.

There are several stereotypes that are used on associations in our security models

The *security_dependency* stereotype is used to indicate that one security concern is dependent on another security concern. For example, authorization is dependent on identity. Any realization of authorization must be accompanied by a realization of identity or there is a design error in the model.

The *applies_to* stereotype is used to relate domain concepts to a security concern and assists in completeness analysis of a design.. For example, the domain concepts of subject, object and privileges apply to the authorization security concern.

The *security_refinement* stereotype is used to indicate a realization of a concept. This type of association is a link between different levels of abstraction. The level is indicated by the *security_abstraction_level* tagged value.

Authentication Models

We have developed models of each of the security concerns. This section shows part of one example for the authentication concern. Space limitations prevent us from showing all models of all the concerns.

The UML diagram in Figure 3 shows part of the model for the authentication concern. This UML class diagram focuses on the supporting concepts for authentication. The «authentication» stereotype in each class is used to provide a classification mechanism.

The supporting domain concept classes have a stereotype of «domain_concept». This gives us another axis of categorization and linkage across models.

The classes are connected by an association with an «applies_to» stereotype. One reads this as an Identity domain concept *applies to* the Authentication concern. The stereotype on the association gives us the possibility of defining rules to be used during analysis. For example, if a solution realization for authentication does not have elements of all four domain concepts, then the realization is defective.

Figure 4 shows a class model of a refinement structure. Each level in the refinement structure is indicated by the integer type tagged value *security_abstraction_level*. The refinements in this structure reflect a decision to realize the authentication concern via a specific mechanism.

The «authentication» stereotype in each class is used to provide a classification mechanism to group these realizations of the concern. There is a different stereotype on the association in this class model. The semantics we give to the «security_refinement» stereotype are those of greater precision of realization of a security concern. This might seem similar to the object-oriented generalization / specialization concepts, but it is different in that we are creating sub-categories of realizations.
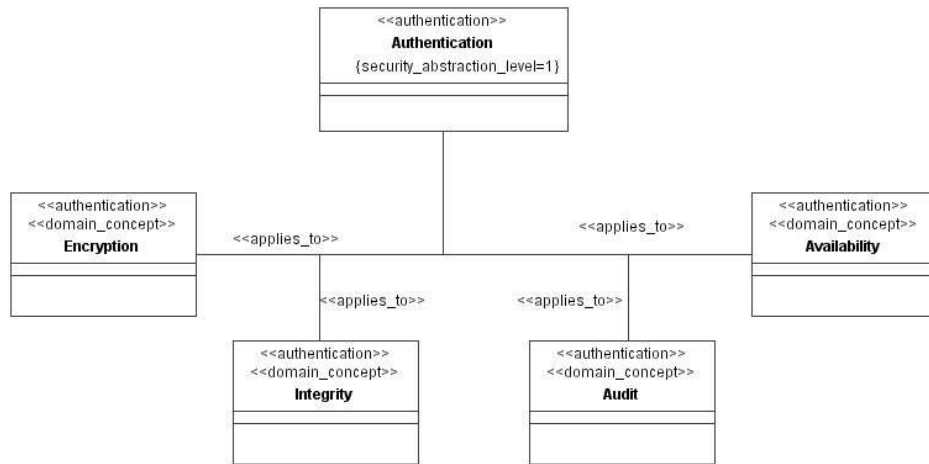
Figure 3 Authentication Concepts UML Model
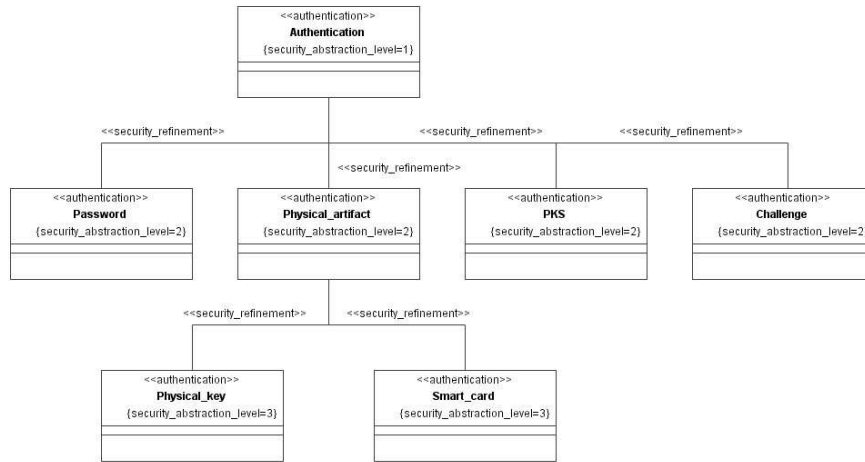
Figure 4 Authentication Refinement

Realization

A repository for holding the models is being developed. This gives a source for the analysis tools and a place for the developer to find this information. The repository will be used to hold the models developed in an AOM approach. Product Data Management (PDM) [39] concepts from the discrete manufacturing industries are being used as the basis for the repository. The component structuring, classification and organization techniques are directly applicable to the situation described here. Dan has over 10 years experience with the construction and use of PDM applications.

5. Conclusions

In this paper we have described an approach for transitioning an imprecise text-based specification of security requirements into a set of patterns expressed in the semi-formal UML notation. The UML expression of the security concerns and their refinements provide a basis for better communication between the design engineer and programmers realizing the design.

We are currently working on a set of tools supporting the different aspects of the AOM program. The tool support covers AOM composition tools, model analysis tools and a repository to manage the models and their evolutions.

The patterns presented in this work are preliminary and further research is needed to establish a stable set of security patterns at various levels of abstraction.