

An Aspect-Based Approach to Modeling Access Control Concerns

Indrakshi Ray, Robert France, Na Li, Geri Georg
Department of Computer Science
Colorado State University
Email: {iray,france,na,georg}@cs.colostate.edu

Keywords: *access control policies, role-based access control, aspects, models*

Contact Address: Indrakshi Ray
Department of Computer Science
Colorado State University
Fort Collins, CO 80523-1873
Email: iray@cs.colostate.edu
Phone: (970) 491-7986, Fax: (970) 491-2466

An Aspect-Based Approach to Modeling Access Control Concerns

Abstract

Organizations define and enforce access control policies to protect sensitive information resources. Access control policies impose additional requirements which ensure that only authorized users have access to the information resources. Addressing an access control policy requirement in a system design often requires one to distribute structures and behaviors pertaining to the policy across design elements. Such policies are said to cross-cut the design. The pervasive nature of access control policies makes it difficult to evolve and analyze them. Ad-hoc creation and evolution of access control policies can result in severely compromised system behaviors that, in turn, can lead to significant loss (such as, financial loss, loss of life, loss of reputation.) In this paper we propose a systematic approach to modeling access control policies during system design. The approach is based on an *aspect-oriented modeling* (AOM) technique that allows system developers to isolate cross-cutting design structures in aspects to support controlled evolution of the structures. In the approach, design structures that represent access control policies are treated as aspects. A policy is incorporated into an system design by composing the aspect with the system design model. The resulting composed model can then be analyzed to determine the impact the policy has on the system design. The AOM approach is illustrated by modeling a basic form of the Role-Based Access Control (RBAC).

1 Introduction

Access control policies are constraints that determine the type of access authorized users have on information resources in order to protect them. For example, an access control policy in a banking system can stipulate that only loan managers can create and update computer-based customer loan accounts. From a software design perspective, access control policies are requirements that must be addressed in a design. Software designers must take into consideration access control policies when they develop models of software design. Access control functionality is often pervasive in that the functionality is spread across the application design. An example of an access control concern that can result in pervasive (or cross-cutting) functionality in a design is an authorization concern that requires all requests for services to be screened to determine whether the requestor can invoke the service. Addressing this concern requires one to design each application service such that an authorization check is carried out before proceeding to carry out its tasks. The pervasive nature of access control functionality is problematic for the following reasons:

- Changes to access control policies are not uncommon and ease of change is desirable. Changing the access control functionality requires making changes in a number of places in the design.
- Evaluating alternative ways of addressing access control concerns is difficult because access control functionality is spread across a system design.
- The pervasive nature of access control functionality makes it difficult to ensure consistency of the access control design elements, especially when different parts of the design are developed by different teams.
- Understanding a pervasive access control functionality can be difficult because the structures and behaviors pertaining to the functionality are not localized in one place.

The above problems can result in software that cannot be trusted to protect sensitive or mission-critical information.

In this paper we propose a systematic approach to modeling and evolving access control design functionality. Our approach is based on *aspect-oriented modeling* (AOM), where an *aspect* is a design unit that localizes cross-cutting functionality. In an AOM approach, a design consists of aspects and a primary model. Each aspect represents a cross-cutting design structure (functionality) that addresses a concern. The primary model is an application design model in which the concerns captured by aspects are not addressed. An application design in which the concerns are addressed is created by composing (weaving) the aspects with the primary model. The composition of aspects and the primary model results in a *woven model*. In this work, access control functionality that is to be incorporated into a software design (the primary model) is treated as an aspect. Each access control aspect is described by a parameterized microarchitecture that imposes the constraints defined by an access control policy. The pervasive functionality is incorporated into a primary model by integrating the aspect with the primary model. Integration involves creating instantiations of the parameterized microarchitecture and composing the instantiations in specified parts of the primary model. The primary and aspect models are composed primarily to support analysis of the interactions between behaviors defined in the aspect and primary models. Such analysis can reveal undesirable emergent behaviors and policy violations.

The localization of cross-cutting access control functionality in aspects supports rigorous specification and analysis, and controlled evolution of access control functionality. The AOM approach also supports controlled evolution of policies. A change in policy can be handled by (1) modifying its design aspect or by (2) replacing the design aspect with another.

In this paper we describe and illustrate how cross-cutting access control functionality can be described by aspects, and how the aspects can be composed with primary models. The rest of the paper is organized as follows. Section 2 describes our approach to modeling cross-cutting access control functionality as aspects. Section 3 illustrates how access control aspects can be composed with primary models. Section 4 gives an overview of related work and Section 5 concludes the paper with an overview of our plans to further develop the approach.

2 Modeling Pervasive Access Control Functionality

In this paper, an aspect is a pattern that captures the structural and behavioral properties that are spread across a design model. Instantiating the pattern and incorporating the instantiations in specified parts of a primary design model produces a design model with the cross-cutting functionality defined by the aspect. An access control aspect is described by a parameterized microarchitecture that shows a pattern of structures and behaviors that reflect constraints defined by an access control policy.

An access control aspect can be modeled from two perspectives: the *structural* perspective and the *dynamic* perspective. The structural perspective identifies the entities constrained by the access control policy and their relationships with each other. The dynamic perspective defines the constraints that the access control policy imposes on behaviors. Our approach to describing access control aspects utilizes the *Unified Modeling Language* (UML) [29]. An access control aspect consists of (1) template forms of UML static structural diagrams (e.g., class diagrams) that represent the microarchitecture from the static structural perspective, and (2) template forms of UML dynamic diagrams (e.g., interaction diagrams) that represent the microarchitecture from the dynamic perspective.

An overview of the AOM approach proposed in this paper is shown in Fig. 1. An aspect-oriented design model consists of aspects and a primary model. In this paper, aspects localize access control functionality. The aspects are integrated with the parts of a primary model that require access control. Integration involves (1) identifying the parts of the primary model that require access control, (2) instantiating the aspects to obtain design views, called *context-specific aspects*, that define how access control will be accomplished in the specified parts, and (3) composing the context-specific aspects (design views) with the primary model. In Fig. 1, the access control aspect is integrated with three parts of the primary model. The woven model produced by composing (weaving) the context-specific aspects and the primary model can then be analyzed to uncover emergent behaviors that result in violations to access control policies. Analysis can also focus on identifying un-

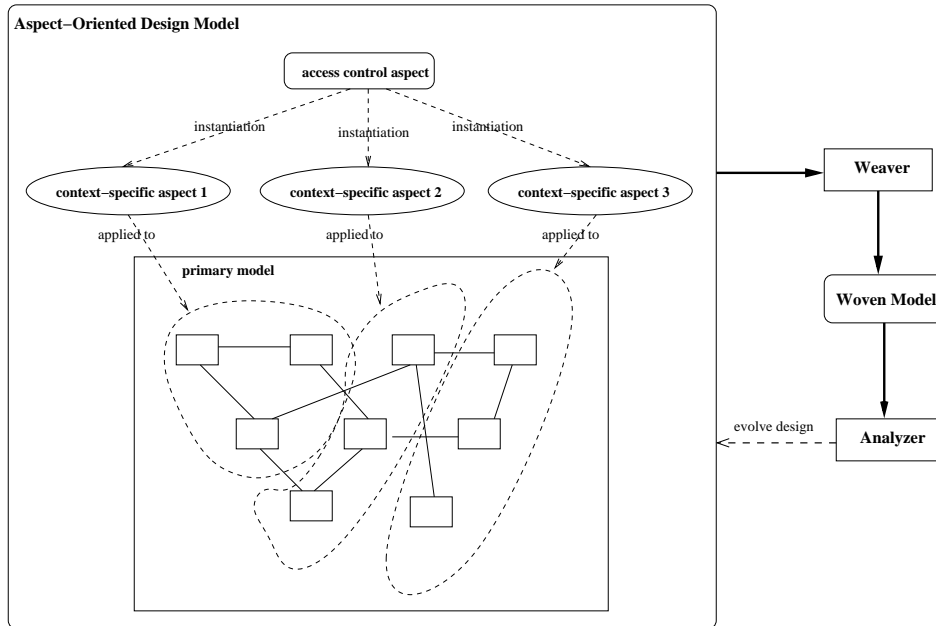


Figure 1: An Overview of the AOM Approach to Modeling Access Control

desirable interactions between access control functionality and other behaviors described in the primary model. Problems uncovered during analysis initiate changes to the design as indicated by the dashed arrow between the analyzer and the design model.

2.1 Modeling Access Control Aspects as Patterns

The structural perspective of an aspect is modeled by a template form of UML static structural diagrams called a *Static Structure Template* (SST). The dynamic perspective is currently described by a template form of UML interaction diagrams called an *Interaction Structure Template* (IST). SSTs and ISTs are based on a pattern specification language described in our previously published papers [20, 21, 22].

The Role Base Access Control (RBAC) model, an access control policy framework, is used to illustrate UML-based formulation of an access control aspect. In this paper, the most basic form of RBAC, referred to as RBAC₀ [54], is modeled. In RBAC₀ permissions are attached to roles and users must be assigned to roles to get the permissions. Permissions determine what operations can be carried out

on resources under access control. A user must establish a session to activate a subset of roles to which the user is assigned. Each user can activate multiple sessions, however, each session is associated with only one user. The operations that a user can perform in a session depend on the roles activated in that session and the permissions associated with those roles.

A model of RBAC₀ is shown in Fig. 2.

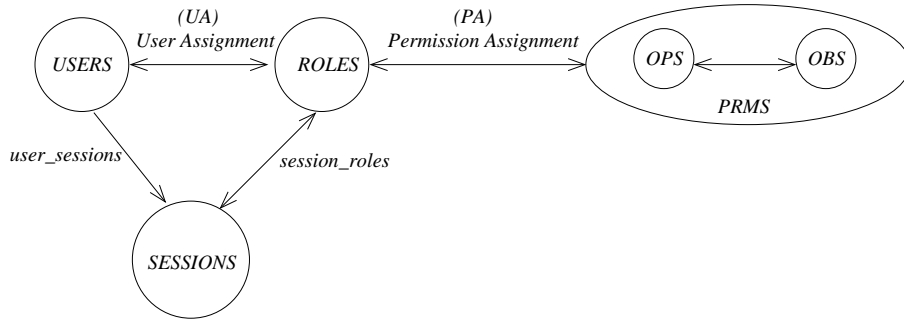


Figure 2: Basic RBAC

Fig. 2 shows that RBAC₀ consists of: 1) a set of users (*USERS*) where a user is an intelligent autonomous agent, 2) a set of roles (*ROLES*) where a role is a job function, 3) a set of objects (*OBS*) where an object is an entity that contains or receives information, 4) a set of operations (*OPS*) where an operation performs tasks, and 5) a set of permissions (*PRMS*) where a permission is an approval to perform operation(s) on object(s). The cardinalities of the relationships are indicated by the absence (denoting one) or presence of arrows (denoting many) on the corresponding associations. For example, the user to session association is one-to-many, indicating that a user can be linked to zero or more sessions at any given time. All other associations shown in the figure are many-to-many.

2.1.1 Modeling Aspects: Structural Perspective

An SST consists of template forms of UML classifiers (e.g., classes) and relationships (e.g., associations). Template classifiers represent parameterized entities, where the parameters can represent class names, class multiplicities, and other class properties. Class templates can also be associated with template forms of attributes and operation respectively called *StructuralFeature templates* and *BehavioralFeature templates*. Instantiating these templates produces attributes and operations. Template relationships represent parameterized relationships, where parameters can represent multiplicity ranges, and relationship names.

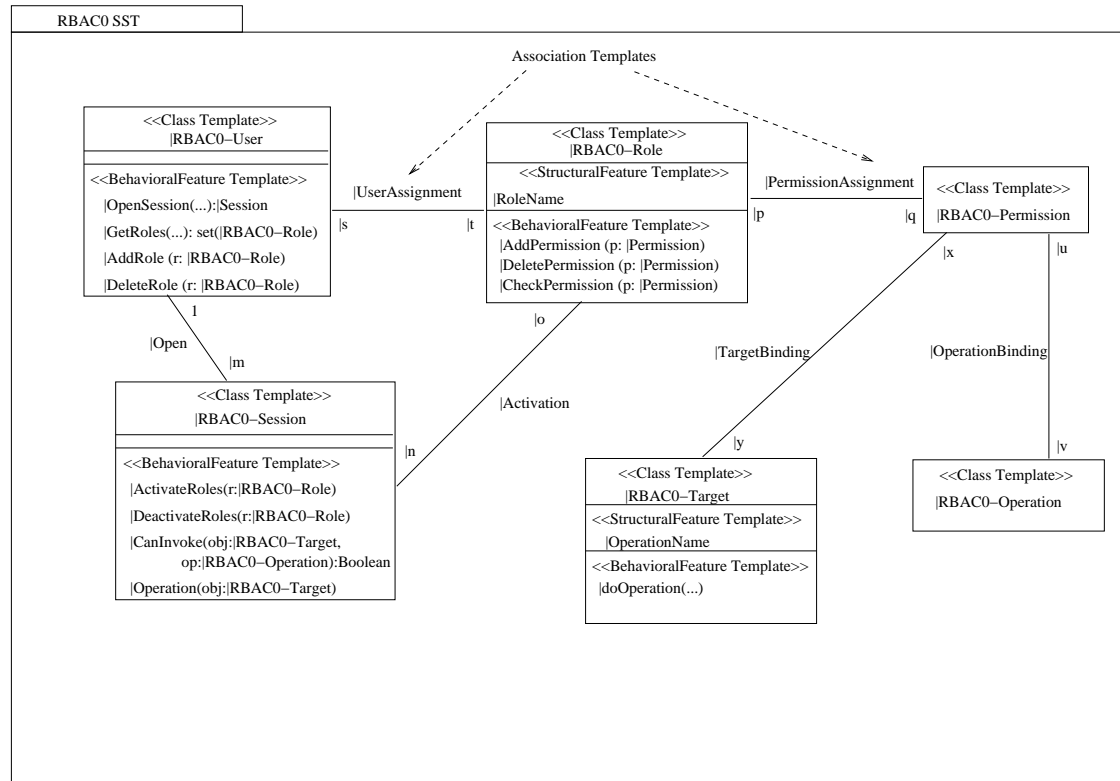


Figure 3: SST for the RBAC0 model

A SST for RBAC₀ is shown in Fig. 3. Template parameters are preceded by |. Substituting UML model elements for these parameters produces a UML static structural diagram that models a context-specific access control microarchitecture from a static perspective. The SST consists of six class templates:

- |RBAC0-User determines UML classes representing users,
- |RBAC0-Role determines UML classes representing user roles,
- |RBAC0-Session determines UML classes representing sessions, and
- |RBAC0-Permission determines permission classes that are associated with target and operation classes characterized by |RBAC0-Target and |RBAC0-Operation templates, respectively.

The class template $|RBAC0-User$ includes BehavioralFeature templates characterizing operations for opening a session ($|OpenSession$), for getting the roles associated with a user ($|GetRoles$), and for adding and deleting user roles ($|AddRole$ and $|DeleteRole$). $|RBAC0-Session$ consists of BehavioralFeature templates characterizing operations for activating ($|ActivateRoles$) and deactivating roles ($|DeactivateRoles$), for checking whether the operation passed as parameter can be invoked or not ($|CanInvoke$), and for performing some operation ($|Operation$). $|RBAC0-Role$ consists of BehavioralFeature templates that characterize operations for adding ($|AddPermission$), deleting ($|DeletePermission$) and checking permissions ($|CheckPermission$). $|RBAC0-Role$ also has a StructuralFeature template called $|RoleName$ that when instantiated with a name yields an attribute representing the name of a role object.

The association templates consist of template parameters for association names and multiplicities. For example, the parameters $|s$ and $|t$ on the $|UserAssignment$ template are multiplicity parameters. Substituting values for the name and the multiplicity parameters in an association template produces a UML association.

Generic constraints can be associated with template elements in an SST. For example, instantiating a BehavioralFeature template that has a template pre- and post-condition yields an operation with a pre- and post-condition. For example, the $|AddRole$ BehavioralFeature template is associated with the following constraint template:

```
context |RBAC0-User:: |AddRole(r:|RBAC0-Role)
pre: self.|RBAC0-Role -> excludes(r)
post: self.|RBAC0-Role ->includes(r)
```

The template determines a family of OCL constraints in which the pre-condition holds true if the role r is not currently linked to the user and the post-condition is true if the role r is linked to the user. To obtain a constraint from the constraint template one simply substitutes values for the template parameters. Class templates can also consist of constraint templates. For example, in RBAC one needs to specify that all the roles activated by a user in a session is a subset of the roles assigned to the user. This is specified by the following constraint template :

```
context |RBAC0-User
self.|RBAC0-Role -> includesAll(self.|Session.|RBAC0-Role)
```

Constraint templates can also be used to constrain the ranges that can be substituted for multiplicity parameters of association templates. For example, the constraint template

$$|u.lowerbound = 1$$

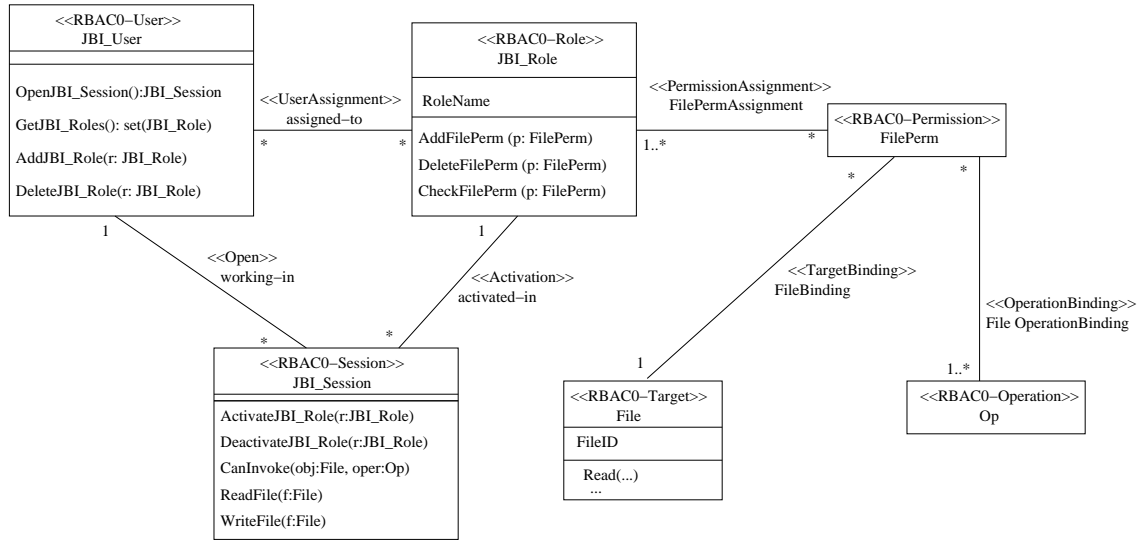


Figure 4: SST for the RBAC0 model

restricts ranges that are substituted for u to have a lower bound of 1.

Given a primary model, a SST can be instantiated to produce a context-specific aspect that describes how access control is addressed in a part of the design model. We use an application, the Joint Battlespace Infosphere (JBI), to illustrate how this can be done. Fig. 4 shows a UML class diagram obtained by substituting values from a JBI primary model for the template parameters in Fig. 3. *JBI_User* working in a *JBI_Session* can carry out a permitted set of file operations. Each session activates *JBI_Roles* that is associated with a set of file operations that are permitted for those activated roles. A *JBI_User* must possess the *JBI_Role* that he can activate in a *JBI_Session*. The class diagram shown in Fig. 4 is a design view that shows only the class attributes and operations needed to model the access control functionality.

Constraint templates are instantiated to create OCL constraints, for example the OCL constraint defining the pre- and post-condition for the *AddJBI_Role* operation for *JBI_User* is obtained by instantiating the *AddRole* constraint template:

```

context JBI_User::AddJBI_Role(r:JBI_Role)
pre: self.JBI_Role -> excludes(r)
post: self.JBI_Role ->includes(r)
    
```

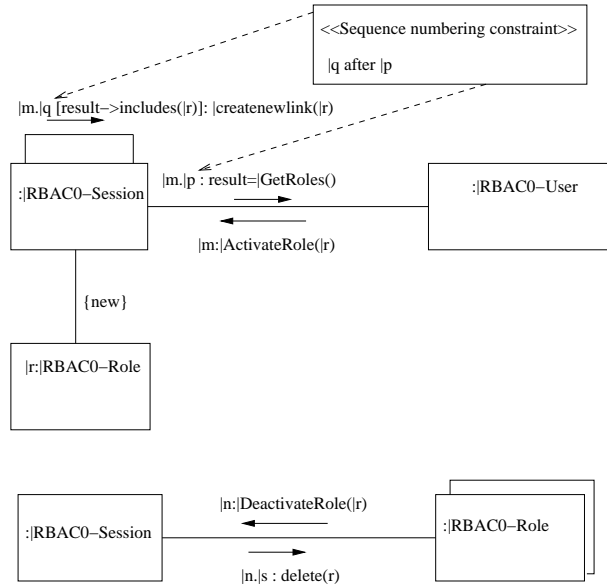


Figure 5: $|ActivateRoles$ and $|DeactivateRoles$ IST

2.1.2 Modeling Aspects: Dynamic Perspective

The interaction view describes the pattern of interactions that take place and is expressed as a set of ISTs. Each IST describes a pattern characterizing a family of scenarios. An IST consists of UML object, collaboration role, link, and message templates.

We show only a few of the ISTs for RBAC to illustrate the form of ISTs. The IST shown in Fig. 5 defines a pattern of interactions that take place when activating and deactivating RBAC roles. An IST consists of template forms of UML collaboration roles and template forms of messages. Message templates have parameters representing sequence expressions and operation calls. For example, $|m.|p : result := |GetRoles()$ is a message template that produces the UML message $1.3 : result := GetMRoles()$ when 1 is substituted for $|m$, 3 for $|p$ and $GetMRoles()$ for $|GetRoles()$.

The invocation of the operation $|ActivateRole(r)$ causes the invocation of the operation $|GetRoles()$ on an $|RBAC0-User$ object. If the user indeed possesses this role, then a new link is created between the corresponding $|RBAC0-Session$ and r objects (indicated by the operation call parameter $|createnewlink(r)$ in the diagram). The constraint $|q$ after $|p$ specifies that the operation call denoted by $|GetRoles()$ occurs before the call denoted by $|createnewlink$.

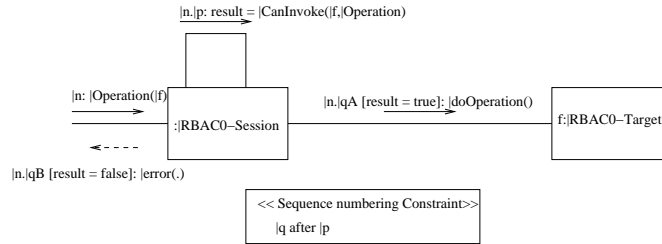


Figure 6: |Operation IST

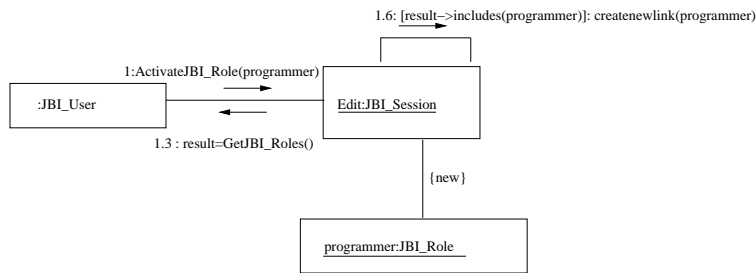


Figure 7: Collaboration Diagram obtained from the |ActivateRoles IST

Fig. 6 shows the IST corresponding to a user invoking an operation in a |RBAC0-Session object on a specified |Target object. This causes the execution of the |CanInvoke(...) operation. The invocation of |CanInvoke(...) returns a boolean value. If the operation returns true, then the specified operation (denoted by |doOperation()) is performed on the |Target object; otherwise, an error message is reported. These two possibilities are indicated by message templates with sequence expressions |n.qA and |n.qB (the A and B indicate that only one of the branches will be taken).

The interaction diagram shown in Fig. 7 is obtained by instantiating the |ActivateRole IST shown in Fig. 5). The interaction diagram shows a |JBI_User object activating the programmer role in an edit session. If the user indeed possesses this role, the programmer role is activated in that session. Note that all the messages are not shown in the collaboration diagram. For example, the interactions labeled

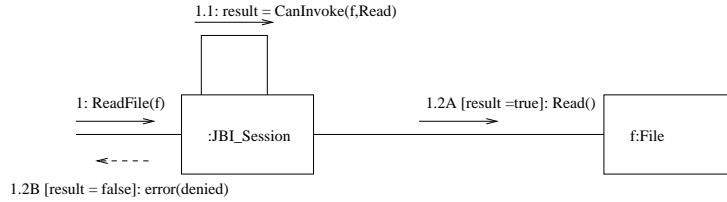


Figure 8: Collaboration Diagram obtained from *Operation IST*

1.1,1.2 and 1.4,1.5 are not shown because they are not part of the view defined by the diagram. The need for sequence gaps in a context-specific policy may arise because of the need to accommodate interactions that address other design concerns. In this example, interactions 1.1,1.2 are required to start a logging activity and 1.4,1.5 are required to log information.

Another collaboration diagram obtained from the *Operation IST* (Fig. 6) is shown in Fig. 8.

3 Composing Aspects with Primary Models

The impact of functionality defined by access control aspects on a primary model is determined by (1) composing (weaving) the access control context-specific aspects with the primary model, and (2) analyzing the woven model. Composing a context-specific aspect with a primary model involves merging the views of the entities and behaviors defined by the aspect with the entity and behavior views defined by the primary model. Composition can involve modifying existing elements in the primary model, adding new model elements to the primary model, and deleting primary model elements. The result of weaving is a new model (the woven model) in which information encapsulated by the aspect is distributed across impacted parts of the primary model.

To illustrate our approach we choose a simple primary model that describes a JBI Project Management system. The static view of the primary model is described by the class diagram shown in Fig. 9. It consists of the class *JBI_User* that describes the users, the class *JBI_Role* that describes the designation and the responsibilities of the project team members, the class *JBI_Session* that describes the operations that can be performed in a session, and the class *File* that describe the files associated with the JBI Project.

The dynamic view of the primary model consists of a set of interaction diagrams. A collaboration diagram of the *ReadFile* operation in the primary model

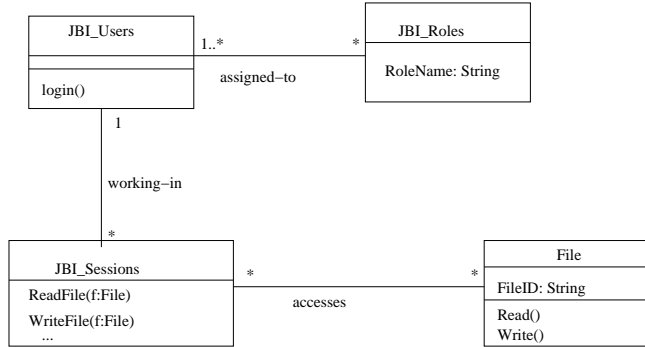


Figure 9: Static View of JBI Project System Primary Model

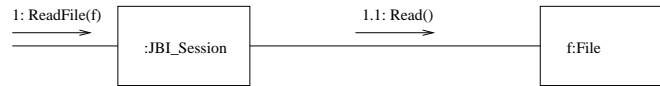


Figure 10: Interaction Diagram in JBI Project System Primary Model (Reading a File)

dynamic view is shown in Fig. 10. This operation does not provide access control to the file. In this section we show how the RBAC access control aspect described in the previous section is composed with this primary model.

In the proposed approach, composing an aspect with a primary model involves the following activities:

1. *Instantiating the aspect to obtain a context-specific aspect:* Before an aspect can be composed with a primary model, the modeler must first obtain context-specific aspects by substituting values for the parameters in the aspect. The substitution values are determined by the locations in the primary model in which access control is desired. If the substitution values are elements in the primary model then the instantiated template elements are *views* of the corresponding primary model elements. If the substitution values are not elements in the primary model, the instantiated template elements are new elements that are to be added to the primary model during composition (these new elements must be associated with at least one existing element in the primary model).
2. *Composing context-specific aspects with the primary model:* The views defined by context-specific aspects are merged with the views defined by the

primary model to obtain a woven model. Elements in the aspect and primary models are merged if and only if they have the same name and are instances of the same metamodel class (i.e., they have the same syntactic types). A model element in an aspect that does not have a matching element in the primary model represents a new model element that is added to the woven model. *Composition directive* can also be provided by the modeler to influence how the composition is carried out. An example of a composition directive is the dominance relationships between matching elements with different behaviors that indicate which elements are to replace matching elements, and specifications of how constraints for matching elements are to be composed. Merging class diagram views of a context-specific aspect and a primary model proceeds as follows (all references to aspect in the following are references to context-specific aspect):

- If the matching elements are classifiers, then the classifier elements are merged. If a classifier element (e.g., an attribute or operation) in the aspect does not match any elements in the matching primary model classifier then the element is added to the classifier in the woven model.
- An operation in the aspect model matches an operation in the primary model if their names and signatures match. For matching operations the modeler should indicate to the weaver, using a composition directive, how the pre- and post-conditions are to be combined (how this is indicated to the weaver is outside the scope of this paper). If this directive is not provided by the modeler, then the default is to combine the preconditions of the matching operations using the logical *or* and combine the postconditions using the logical *and*.
- An attribute in the aspect model matches an attribute in the primary model if their names and types match. If the matching attributes are associated with constraints, then the modeler should indicate to the weaver, using a composition directive, how constraints are to be combined. The default is to connect the constraints using a logical *and*.
- A relationship in an aspect matches a relationship in the primary model if it has the same name. If matching associations have different multiplicities or role names, then the modeler must indicate to the weaver, using a composition directive, which view should dominate.

Merging of interaction views involves (1) matching aspect collaboration roles with primary model collaboration roles, (2) adding aspect collaboration roles that do not match primary model collaboration roles to the woven model, and (3) merging the messages specified in the views based on composition directives provided

by the modeler. The last step requires the modeler to define dominance or sequence number relationships between messages in the primary and aspect views.

We illustrate the above activities using the JBI primary model and RBAC aspect.

Example 1 The RBAC aspect is illustrated in Fig. 3, Fig. 5 and Fig. 6.

Activity 1 is concerned with obtaining context-specific aspects from the RBAC aspect. In this example, the RBAC aspect is to be applied in one place (the entire model), thus only one context-specific aspect is generated. The static view of this context-specific aspect is shown in Fig. 4. The substitutions used for the classifiers and associations are indicated by the stereotypes. The operations *ReadFile* and *WriteFile* in the primary model class *JBI_Session* are used as substitution values for the operation template *|Operation* in *|RBACO – Session* (i.e., this template is instantiated twice in this case). Similarly the *|doOperation* template in *|RBACO – Target* is instantiated twice with the values *Read(...)* and *Write(...)*. The other substitutions for the attributes and operations should be evident from the names used. Some of the interaction diagrams in the dynamic view of the context-specific aspects are shown in Fig. 7 and Fig. 8.

The next step is to merge the context-specific aspect with the primary model to obtain the woven model. Comparing the class diagrams of the primary model (Fig. 9) and the class diagram for the context-specific policy (Fig. 4) we observe the following:

1. The classes *FilePerm* and *Op* are not in the class diagram of the primary model.
2. The associations *FilePermAssignment*, *FileBinding*, *FileOperationBinding*, and *activated-in* shown in Fig. 4 are not in the class diagram of the primary model.
3. The classes *JBI_Users*, *JBI_Roles*, *JBI_Sessions* have operations pertaining to RBAC that are absent in the primary model.
4. Operations, such as, *ReadFile*, *WriteFile* are present in both the primary model and also in the model describing the context-specific policy. In this case the intent is that the definition given in the aspect replaces the definitions given in the primary model. This *composition directive* must be provided by the modeler.

The weaving operation takes these observations into account and produces the following woven model:

1. The missing classes *FilePerm* and *Op* are added in the woven model.
2. The associations *FilePermAssignment*, *FileBinding*, *FileOperationBinding*, *activated-in* are added to the woven model.
3. New operations pertaining to RBAC (such as, *AddJBI_Role*, *AddFilePerm*) are added to the classes *JBI_Users*, *JBI_Roles*, and *JBI_Sessions*.
4. The operations *ReadFile* and *WriteFile* are modified in the woven model (the operations specified in the aspect dominates the operations specified in the primary model).

Fig. 11 gives the woven static diagram. Note that this is identical to Fig. 4. This is because the access control policy constraints over-ride the constraints in the primary model. In cases where we have to weave in multiple context-sensitive policies, the woven model will not be identical to the class diagrams of a specific context-sensitive policy.

The weaving of the collaboration diagrams shown in Fig. 10 and Fig. 8 is determined by a composition directive provided by the modeler that states that the aspect model replaces the diagram in the primary model.

4 Related Work

This work is related to two distinct research areas: policy specification and aspect-oriented development. Consequently, we devote a section to each of these research areas.

4.1 Security Policies

In this section we briefly mention some work on security policies. Damianou's thesis [15] provides a comprehensive survey about the important works in this area. A large volume of research exists in the area of security policies. Some of these focus on access control models [3, 6, 7, 9, 13, 26, 43, 53, 55], some on specification of access control policies [4, 5, 10, 12, 16, 30, 31, 33, 35, 47, 46, 51], and others on analyzing conflicts of security policies [28, 40, 41, 42, 56, 57]. For lack of space, we mention a few works on policy specification languages.

Formal logic-based approaches [4, 5, 10, 12, 30, 35, 47] are often used to specify security policies. Jajodia et al. [35] propose an authorization specification language (ASL) based on stratified clause form logic. Both negative and positive authorizations can be expressed using this logic. The language also includes *integrity rules* that can be used to specify application-dependent conditions that limit

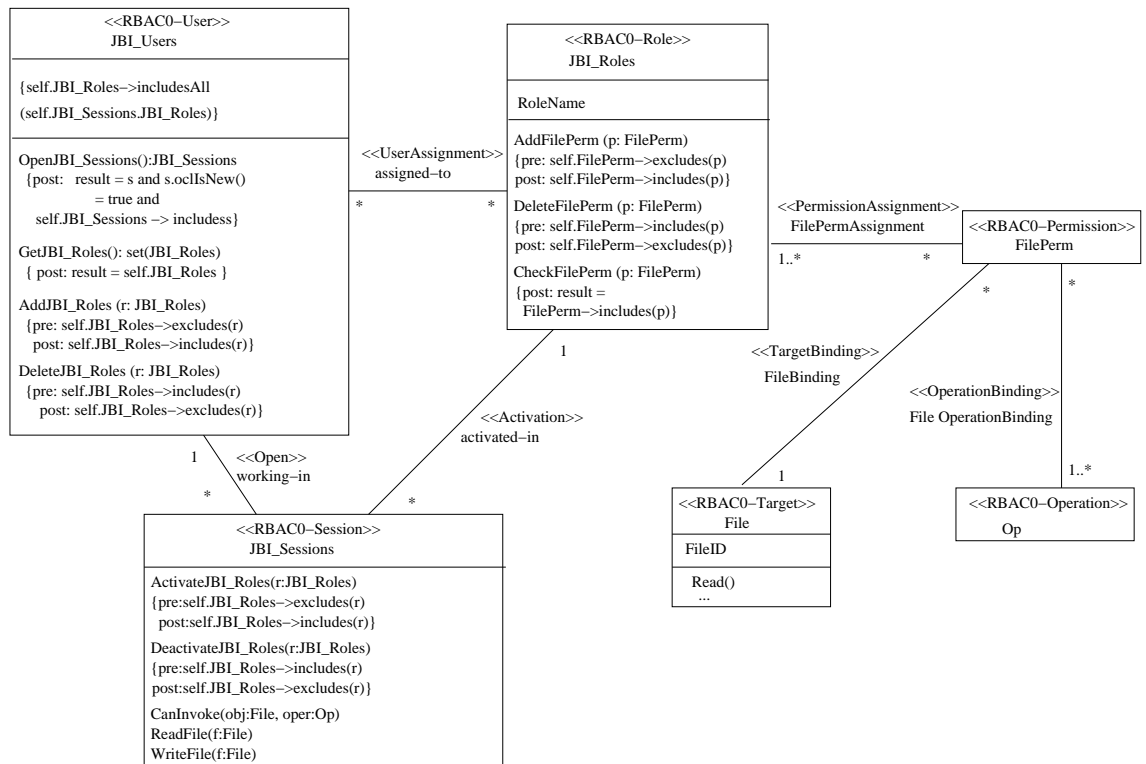


Figure 11: Woven Static Model of RBAC and Primary

the range of acceptable access control policies. This language provides support for role based access control but no direct support for delegations or obligations. Barker [4] also uses stratified clause form logic to express access control policies with special attention to RBAC. In a subsequent work [5] the authors show how policies specified in stratified logic can be translated into SQL to protect a relational database from unauthorized read and update requests. Ortalo [47] describe a language for specifying security policies based on deontic logic. Researchers [30] at the Cambridge University have defined a language called Role Definition Language (RDL) based on Horn clauses. RDL is based on a set of rules that indicate the conditions under which a client may obtain a name or role. The conditions for obtaining a role depend on the credentials of the client. The notion of delegation in RDL is different in the sense that roles and not access rights are delegated. A client may delegate a role that he himself does not possess. Chen et al. [12] propose a language based on set theory for specifying RBAC state related constraints. Bertino et al. [10] extends the RBAC model with a temporal model called TRBAC. The language proposed by Bertino can specify periodic activation and deactivation of roles using periodic expressions. They can also specify temporal dependencies among role activation and deactivation using role triggers. Formal logic-based approaches, although, useful for analyzing security policies, are relatively difficult to implement.

Other researchers have used high-level languages to specify policies [51, 31, 46, 33]. Although high-level languages are easier to understand than formal logic-based approaches, they are not analyzable. We briefly mention some important works in this area. Ribeiro et al. [51] propose a Security Policy Language (SPL) for specifying authorization and obligation policies. Policies are specified using constraint rules. Tower [31] is a language for specifying RBAC policies. The policies are specified using *objects, privileges, permissions, users, and roles*. Privileges define a specific access type on an object, permissions are composed of privileges, and roles contain a set of permissions. In addition privileges can also be associated with *conditions* and *actions*. *Conditions* limit the applicability of the privilege. *Actions* are executed when methods associated with the privileges are invoked. The *Organization for the Advancement of Structured Information Standards* (OASIS) technical committee advocates the use of XML for expressing access control policies [46]. They proposed XACML which is an XML specification for expressing policies for information access over the Internet. The policy specification in XACML is very verbose and not aimed for human interpretation. LaSCO [33] is a graphical approach for specifying policies. The graphical format of LaSCO helps in human interpretation but is not very expressive. Ponder [16] is a specification language that allows various kinds of policies, such as, authorization, obligation, and delegation policies to be specified. Policies are specified in terms of *subject-*

domain, *target-domain*, and *access-lists*. The subject-domain specifies the set of subjects that can perform the operations specified in the access-lists on the objects in the target-domain. The authors have also developed a toolkit for policy specification and deployment [17]. Steen et al. [59] propose a new language for expressing policies that can be applied over an enterprise that is modeled using UML. The language contains embedded OCL constraints. The constraints cannot specify activation/deactivation of roles or assignment of users or permissions to roles. It also does not allow for the composition of policies.

4.2 Aspect-Oriented Modeling

There has been much work on aspect-oriented programming (AOP) [8, 38, 39, 48, 60]. A number of authors have tackled the problem of defining and weaving aspects above the programming language level (e.g., see [14, 19, 20, 27, 45, 49, 58]). In the latter cases, aspect specifications are often templates, and they are generally woven by using regular expressions to match primary model elements and aspect elements. Aspect composition in these approaches usually involves wrapping additional functionality around a primary model. Proper naming and structuring of model elements is required to apply the aspect, so it is conceivable that effort must be applied to re-factor existing models to correctly compose them with aspect models. The property-oriented approach used in our aspect-oriented modeling (AOM) supports a more flexible and rigorous approach to aspect definition and weaving in which primary models do not need to have syntactically-equivalent names or structures in order to have aspects woven into them.

There has been some work on modeling dependability concerns using the UML, in particular work on modeling security concerns (e.g., see [1, 2, 11, 18, 21, 25, 34, 37, 36, 44, 61, 62]). These works use the UML extension mechanisms to introduce representations of dependability concerns in UML models. For example, Chan and Kwok [11] provide a design pattern for security that addresses asset and functional distribution, vulnerability, threat, and impact of loss. UML stereotypes identify classes that have particular security needs due to their vulnerability either as assets or as a result of functional distribution. Jurjens [36, 37] models security mechanisms based on the multi-level classification of data in a system using an extended form of the UML called UMLsec. The UML tag extension mechanism is used to denote sensitive data. Statechart diagrams model the dynamic behavior of objects, and sequence diagrams are used to model protocols. These works are significant and complementary to our work on AOM in that they illustrate how the UML can be extended to directly represent dependability concerns. An AOM weaving process can be designed to produce extended forms of the UML that reflect the properties expressed in the aspects in a more direct manner.

In our previous works [20, 24] we show how aspects can be modeled as structural and behavioral patterns, and how aspects can be woven into designs expressed in the UML. We show in [24] how security concerns can be encapsulated and then woven with models of system functionality. In [23] we model two independent security mechanisms as aspects and show how these two aspects (authentication and auditing) can be woven in with a primary model. We also show that the order in which the aspects are woven is important. An incorrect weaving order will produce a woven model that does not meet design goals. The work on UMLAUT [32] is similar to our approach but they use meta-collaboration roles instead of templates.

5 Conclusion

In this paper we describe how cross-cutting access control functionality can be localized in aspects to ease evolution of policies and design. Aspects are described by generic microarchitectures that can be applied across and within applications. Instantiating an aspect results in a context-specific aspect that can be merged with a specified part of a primary design model. UML diagram templates are used to express aspects. This enables the use of aspects in UML-based software modeling and leverages existing tool support for the UML.

We also show how aspects can be composed with a primary model to produce a woven model that can be analyzed to identify undesirable emergent behaviors. Composition can be influenced by composition directives provided by the modeler. The composition results in a woven model that describes the application incorporating the access control functionality.

In this paper we focused on defining and composing a single access control aspect with a primary model. Sometimes there is a need to compose more than one aspect into a primary model. The idea presented in this paper can be extended for access control policy composition. Suppose we want to compose multiple aspects P_1, P_2, \dots, P_n with a primary model. We first compose the access control policy P_1 with the primary model. The woven model then becomes a new primary model, which is then composed with aspect P_2 . This approach is repeated until all the aspects have been woven.

We are currently developing an analysis technique that involves verifying the woven model against a select set of representative scenarios. The scenarios can represent malicious attacks or authorized interactions. Malicious attack scenarios are used to determine if the constraints specified in policies are sufficient to prevent the attacks from compromising protected resources, while the authorized interaction scenarios are used to determine that authorized activities are not restricted by the constraints. We will also investigate the use of model-checking techniques for

statically analyzing dynamic aspects of woven models.

References

- [1] G.J. Ahn and M. E. Shin. Role-based authorization constraints specification using object constraint language. In *Proceedings of the 10th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '01)*, pages 157–162, Cambridge, Massachusetts, June 2001.
- [2] H. A. Ali. A new model for monitoring intrusion based on Petri nets. *Information Management and Computer Security*, 9(4):175–182, 2002.
- [3] R. J. Anderson. A Security Policy Model for Clinical Information Systems. In *IEEE Symposium on Security and Privacy*, pages 30–43, Oakland, CA, May 1996.
- [4] S. Barker. Security Policy Specification in Logic. In *Proceedings of the International Conference on Artificial Intelligence*, pages 143–148, Las Vegas, NV, 2000.
- [5] S. Barker and A. Rosenthal. Flexible Security Policies in SQL. In *Proceedings of the 15th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, Niagara-on-the-Lake, Canada, 2001.
- [6] J. F. Barkley, K. Beznosov, and J. Uppal. Supporting Relationships in Access Control Using Role Based Access Control. In *Proceedings of the 4th ACM Workshop on Role-Based Access Control*, pages 55–65, Fairfax, VA, October 1999.
- [7] D. E. Bell and L. J. LaPadula. Secure computer system: Unified exposition and multiple interpretations. Technical Report MTR-2997, MITRE Corporation, Bedford, MA, July 1975.
- [8] L. Bergmans and M. Aksit. Composing multiple concerns using composition filters. *Communications of the ACM*, 44(10), Oct 2001.
- [9] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An Access Control Model Supporting Periodicity Constraints and Temporal Reasoning. *ACM Transactions on Database Systems*, 23(3):231–285, 1998.

- [10] E. Bertino, P. Bonatti, and E. Ferrari. TRBAC: A Temporal Role-Based Access Control Model. In *Proceedings of the 5th ACM Workshop on Role-Based Access Control*, pages 21–30, Berlin, Germany, 2000.
- [11] M. T. Chan and L. F. Kwok. Integrating security design into the software development process for e-commerce systems. *Information Management and Computer Security*, 9(2-3):112–122, 2001.
- [12] F. Chen and R. Sandhu. Constraints for Role-Based Access Control. In *Proceedings of the 1st ACM Workshop on Role-Based Access Control*, Gaithersburg, MD, 1995.
- [13] D. D. Clark and D. R. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *IEEE Symposium on Security and Privacy*, Oakland, CA, May 1987.
- [14] S. Clarke and J. Murphy. Developing a tool to support the application of aspect-oriented programming principles to the design phase. In *Proceedings of the International Conference on Software Engineering (ICSE '98)*, Kyoto, Japan, April 1998.
- [15] N. Damianou. *A Policy Framework for Management of Distributed Systems*. PhD thesis, University of London, London, U.K., 2002.
- [16] N. Damianou and N. Dulay. The Ponder Policy Specification Language. In *Proceedings of the Policy Workshop*, Bristol, U.K., 2001.
- [17] N. Damianou, N. Dulay, E. Lupu, M. Sloman, and T. Tonouchi. Tools for Domain-based Policy Management of Distributed Systems. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*, Florence, Italy, April 2002.
- [18] Z. Diamadi and M. J. Fischer. A simple game for the study of trust in distributed systems. *Wuhan University Journal of Natural Sciences*, 6(1-2):72–82, 2001.
- [19] J. L. Fiadeiro and A. Lopes. Algebraic semantics of co-ordination or what is it in a signature? In A. Haeberer, editor, *Proceedings of the 7th International Conference on Algebraic Methodology and Software Technology (AMAST'98)*, volume 1548 of *Lecture Notes in Computer Science*, pages 293–307, Amazonia, Brasil, January 1999. Springer-Verlag.

- [20] R. France and G. Georg. Modeling fault tolerant concerns using aspects. Technical Report 02-102, Computer Science Department, Colorado State University, 2002.
- [21] R. France, D. Kim, E. Song, and S. Ghosh. Using Roles to Characterize Model Families. In Haim Kilov, editor, *Practical foundations of business and system specifications*. Kluwer Academic Publishers, 2002.
- [22] R. B. France, D. K. Kim, and E. Song. Patterns as precise characterizations of designs. Technical Report 02-101, Computer Science Department, Colorado State University, 2002.
- [23] Geri Georg, Robert France, and Indrakshi Ray. Designing High Integrity Systems using Aspects. In *Proceedings of the Fifth IFIP TC-11 WG 11.5 Working Conference on Integrity and Internal Control in Information Systems (IICIS 2002)*, Bonn, Germany, November 2002.
- [24] Geri Georg, Indrakshi Ray, and Robert France. Using Aspects to Design a Secure System. In *Proceedings of the Interational Conference on Engineering Complex Computing Systems (ICECCS 2002)*, Greenbelt, MD, December 2002. ACM Press.
- [25] L. Giuri and P. Iglio. A role-based secure database design tool. In *Proceedings of the 12th Annual Computer Security Applications Conference*, pages 203–212, 1996.
- [26] V. Gligor. Characteristics of Role Based Access Control. In *Proceedings of the 1st ACM/NIST on Role-Based Access Control Workshop*, Gaithersburg, MD, November 1995.
- [27] J. Gray, T. Bapty, S. Neema, and J. Tuck. Handling crosscutting constraints in domain-specific modeling. *Communications of the ACM*, 44(10):87–93, October 2002.
- [28] N. Griffeth and H. Velthuijsen. Reasoning About Goals to Resolve Conflicts. In *Proceedings of the International Conference on Intelligent Cooperative Information Systems*, pages 197–204, Los Alamitos, California, 1993.
- [29] The Object Management Group. The Unified Modeling Language. Version 1.4, OMG, formal/2001-09-67, 2001.
- [30] R. J. Hayton, J. M. Bacon, and K. Moody. Access Control in Open Distributed Environment. In *IEEE Symposium on Security and Privacy*, pages 3–14, Oakland, CA, May 1998.

- [31] M. Hitchens and V. Varadarajan. Tower: A Language for Role-Based Access Control. In *Proceedings of the Policy Workshop*, Bristol, U.K., 2001.
- [32] W.M. Ho, F. Pennaneac'h, and N. Plouzeau. UMLAUT: A framework for weaving UML-based aspect-oriented designs. In *Proceedings of the Technology of object-oriented languages and systems conference (TOOLS Europe)*, volume 33, pages 324–334. IEEE Computer Society, 2000.
- [33] J. A. Hoagland, R. Pandey, and K. N. Levitt. Security Policy Specification Using a Graphical Approach. Technical Report CSE-98-3, Computer Science Department, University of California Davis, July 1998.
- [34] R. Holbein, S. Teufel, and K. Bauknecht. A formal security design approach for information exchange in organisations. In *Proceedings of the 9th Annual IFIP WG 11.3 Working Conference on Database Security*, pages 267–285, Rennselaerville, New York, August 1995.
- [35] S. Jajodia, P. Samarati, and V. S. Subrahmanian. A Logical Language for Expressing Authorizations. In *IEEE Symposium on Security and Privacy*, pages 31–42, Oakland, CA, May 1997.
- [36] J. Jurjens. Modeling audit security for smart-card payment schemes with UMLsec. In *Proceedings of the IFIP TC11 16th International Conference on Information Security (IFIP/Sec'01)*, pages 93–107, Paris, France, June 2001.
- [37] J. Jurjens. Towards development of secure systems using UMLsec. In *Proceedings of the 4th International Conference on Fundamental Approaches to Software Engineering (FASE '01)*, volume 2029 of *Lecture Notes in Computer Science*, pages 187–200, Genova, Italy, April 2001.
- [38] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold. Getting started with AspectJ. *Communications of the ACM*, 44(10):59–65, October 2001.
- [39] K. Kieberherr, D. Orleans, and J. Ovlinger. Aspect-oriented programming with adaptive methods. *Communications of the ACM*, 44(10):39–41, October 2001.
- [40] E. Lupu and M. Sloman. Conflict Analysis for Management Policies. In *Proceedings of the 5th IFIP/IEEE International Symposium on Integrated Network Management*, pages 430–443, San Diego, California, May 1997. Chapman & Hall.

- [41] J. Michael. *A Formal Process for Testing Consistency of Composed Security Policy*. PhD thesis, George Mason University, Fairfax, Virginia, 1993.
- [42] N. Minsky, V. Ungureanu, W. Wang, and J. Zhang. Building Reconfiguration Primitives into the Law of a System. In *Proceedings of the International Conference on Configurable Distributed Systems*, pages 89–97, Annapolis, MD, May 1996.
- [43] J. D. Moffett. Control Principles and Role Hierarchies. In *Proceedings of the 3rd ACM/NIST on Role-Based Access Control Workshop*, Fairfax, VA, October 1998.
- [44] U. Nerurkar. A strategy that's both practical and generic. *Dr. Dobbs's Journal*, 25(11):50–56, November 2001.
- [45] P. Netinant, T. Elrad, and M. E. Fayad. A layered approach to building open aspect-oriented systems. *Communications of the ACM*, 44(10):83–85, October 2001.
- [46] OASIS. XACML Language Proposal, Version 0.8. Technical report, Organization for the Advancement of Structured Information Standards, January 2002. Available electronically from <http://www.oasis-open.org/committees/xacml>.
- [47] R. Ortalo. A Flexible Method for Information Systems Security Policy Specification. In *Proceedings of the 5th European Symposium on Research in Computer Security*, Louvain-la-Neuve, Belgium, 1998. Springer-Verlag.
- [48] H. Ossher and P. Tarr. Using multidimensional separation of concerns to (re)shape evolving software. *Communications of the ACM*, 44(10):43–50, October 2001.
- [49] J. A. D. Pace and M. R. Campo. Analyzing the role of aspects in software design. *Communications of the ACM*, 44(10):66–73, Oct. 2001.
- [50] T. Reenskaug, P. Wold, and O. A. Lehne. *Working with Objects: The OORAM Software Engineering Method*. Manning/Prentice Hall, 1996.
- [51] C. Ribeiro, A. Zuquete, and P. Ferreira. SPL: An Access Control Language for Security Policies with Complex Constraints. In *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA, February 2001.

- [52] D. Riehle and T. Gross. Role Model Based Framework Design and Integration. In *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '98)*, pages 117–133, Vancouver, Canada, October 1998. ACM Press.
- [53] P. Samarati and S. Vimercati. Access Control: Policies, Models and Mechanisms. In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design (Tutorial Lectures)*, pages 137–196, September 2000.
- [54] R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996.
- [55] R. Sandhu, D. Ferraiolo, and R. Kuhn. The NIST Model for Role-Based Access Control: Towards a Unified Standard. In *Proceedings of the 5th ACM Workshop on Role-Based Access Control*, pages 47–61, Berlin, Germany, July 2000.
- [56] E. Sibley. Experiments in Organizational Policy Representation: Results to Date. In *Proceedings of the IEEE International Conference on Systems Man and Cybernetics*, pages 337–342, Los Alamitos, CA, 1993. IEEE Computer Society Press.
- [57] E. Sibley, J. Michael, and R. Wexelblat. Use of an Experimental Policy Workbench: Description and Preliminary Results. In C. Landwehr and S. Jajodia, editors, *Database Security V: Status and Prospects*, pages 47–76. Elsevier Science Publishers, 1992.
- [58] A. R. Silva. Separation and composition of overlapping and interacting concerns. In *OOPSLA '99 First Workshop on Multi-Dimensional separation of Concerns in Object-Oriented Systems*, Denver, Colorado, November 1999.
- [59] M. W. A. Steen and J. Derrick. Formalizing ODP Enterprise Policies. In *Proceedings of the 3rd International Enterprise Distributed Object Computing Conference*, Mannheim, Germany, September 1999.
- [60] G. T. Sullivan. Aspect-oriented programming using reflection and metaobject protocols. *Communications of the ACM*, 44(10):95–97, October 2001.
- [61] D. Trcek. Security policy conceptual modeling and formalization for networked information systems. *Computer Communications*, 23(17):1716–1723, 2000.
- [62] J. J. Whitmore. A method for designing secure solutions. *IBM Systems journal*, 40(3):747–768, 2001.