UACML: Unified Access Control Modeling Language

Nadera Slimani, Hemanth Khambhammettu, Kamel Adi, Luigi Logrippo
Department of Computer Science and Engineering
Université du Québec en Outaouais, Canada
{slin02, khamhe01, kamel.adi, luigi.logrippo}@uqo.ca

Abstract

Incorporating security requirements into system design models is receiving increasing interest. Access control requirements are an important part of overall system security requirements. Existing approaches that incorporate access control requirements into system design models have directly been developed on top of specific access control models. In these approaches, there exists a *tight-coupling* between the modeling language and underlying access control model(s) on which the modeling language is developed. Consequently, these approaches can *only* support security requirements for the access control model(s) on which they were developed.

We propose an alternative approach in this work by adopting a "*metamodel of access control*" as a basis for developing a UML-based modeling language. The usage of a metamodel of access control offers at least two benefits: (i) our modeling language is able to represent a variety of access control requirements in a generic way and (ii) our modeling language is independent of specific access control models. By using examples, we demonstrate that our approach is useful for developing a generic modeling language of access control that is *simple*, yet *powerful* for representing a variety of access control models.

1 Introduction

Model driven architecture (MDA) is a well known "model-centric" approach for supporting the software development process [8]. *Model Driven Security* (MDS) is a methodology, based on the MDA approach, for developing secure systems [2]. MDS allows designers to specify system models along with their security requirements and use tools to automatically generate system architectures from system models. An important advantage of using the MDS approach is the development of secure system models that are both *reusable* and *evolvable*.

Access control is the means to enable or restrict the ability of subjects to access protected resources. Typically, a subject may be a user, process or program. Access control policies are an important component of information security policies, which ultimately translate into access control mechanisms [5]. An *access control policy* specifies high-level rules according to which a system must govern access to its protected resources.

Over the last four decades, various approaches to access control have been developed. The following are three such approaches to access control which have been widely discussed in the literature.

- *Discretionary* access controls (DAC) governs access to protected resources based on the identity of users or groups to which they belong [16].

- *Mandatory* access control (MAC) enforces access control on the basis of regulations mandated by a central authority [14].

  The Bell-LaPadula model [3] is the best known MAC model for enforcing information flow controls.

- *Role-based* access control (RBAC) offers an efficient way for representing organizational structures and also provides the basis for an efficient access control mechanism that simplifies security administration [15]. Hence, role-based models are appropriate for enforcing policies based on organization job functions and are most useful to simplify administration.

The access control requirements of organizations have become increasingly complex in recent years, making access control mechanisms based on a *single* (traditional) access control model either inefficient or inappropriate.

Consequently, modern access control systems typically require that access control requirements be specified (and enforced) by using a combination of access control models. We refer to policies that are based on combination of access control models as *"hybrid"* access control policies.

We use the *unified modeling language* (UML) that is a widely used language for visual modeling [11]. UML is a well founded language with a set of relevant diagrams and their associated concepts for modeling systems. UML has become the de-facto standard for building object-oriented software. We believe the current UML specification is ready to be extended to support access control systems design. In addition, system designers are able to validate UML-diagrams by using a dedicated and formal language, called *object constraint language* (OCL) [10].

The aim of our work is to develop a UML-based visual modeling language that provides support for a wide range of access control models in a "generic" way and still is as simple as possible. The key idea of our work is to use a "metamodel of access control" for the modeling language that we propose.

Adopting a metamodel of access control as a basis for our modeling language offers at least two benefits: (i) it *simplifies* the syntax of our modeling language and (ii) it provides policy authors with a *general framework* for representing a variety of access control requirements [1]. Consequently, the modeling language that we develop by using a metamodel of access control will be a simple, yet powerful, language to model several different types of access control policies; both individually and combined.

Recently, Barker proposed a metamodel of access control [1]. However, Barker's metamodel does not provide support for resource hierarchies and action hierarchies, which are useful for specifying high-level access rules. In order to better maintain resources and their data, resources are often organized in hierarchies, where a resource may contain another resource(s). Furthermore, in order to simplify security administration, it is often desirable to provide support for policies by which authorization to access a resource implies an authorization to access all its sub-resources.[1] Consider, for example, a user u who has a privilege `Full_Access` for subject ***`Computer Science`*** of a digital library. Then, u will *implicitly* be authorized for privilege `Full_Access` to all "subdisciplines" of subject ***`Computer Science`***,

[1]Several commercial software products, the IBM Websphere Portal [4], for example, provide support for resources to be organized in a hierarchical structure, such that the access control configuration of a given resource is propagated to all of its child resources.

such as *Bioinformatics*, *Software Engineering*, and *Security and Cryptology*.

As well, action hierarchies have been shown to be useful for generating high-level access rules by defining composite actions [2]. The semantics of a composite action state that the right to perform an action implies the right to perform any of the (transitively) contained subordinate actions. For example, a composite action `Full_Access` of a 'digital library' may imply actions `on_screen_read`, `download` and `print` all parts of documents stored in the repository, whereas an action `Guest_Access` may *only* imply action `on_screen_abstract_read`.

Rather than developing yet another metamodel of access control, we *extend* the metamodel of access control proposed by Barker to provide support for *resource hierarchies* and *action hierarchies*. Essentially, our work proposes a "generic" modeling language for access control through a metametamodel that can be instantiated to derive metamodels, which will specifically represent well known access control models, such as MAC and RBAC, and also combinations of access control models for supporting hybrid access control models.

The rest of the paper is organized as follows. In Section 2, we describe the motivating example to our work. Our UML-based modeling language for access control is described in Section 3. In Section 4, we compare the contributions of this paper against the most representative work of literature within the domain of UML-based modeling languages for access control. Section 5 summarizes the contributions of the paper and outlines future directions.

2 Motivation

In this section, we motivate the need to develop a generic modeling language that is based on a metamodel of access control. We present an example of a scenario that requires the specification of a hybrid access control policy. Consider, for instance, that subsequent to receiving a *request for proposal* (RFP), staff within an organization prepare and submit a "bid" for some (proposed) work to be completed. Let us assume that the following three tasks must be performed for preparing a bid.

Task T_1**:** Provide relevant inputs for the work to be completed within a `Input_RFP` document.

Task T_2**:** Combine the inputs provided and prepare a *response to RFP* (`resp_RFP`) document.

Task T_3**:** Prepare a *bid proposal* (`Bid_RFP`) document by appending *financial information*, such as billing and costs, to the `resp_RFP` document.

Further, the policy of the organization may require that the following rules be enforced while performing the above three tasks.

Rule R_1**:** Only members of the concerned project (say, `Project 1`) must be able to access the above documents.

Rule R_2**:** Every member of the project must perform task T_1.

Rule R_3**:** Tasks T_2 and T_3 must be performed by a team leader and a manager, respectively.

Rule R_4**:** Only certain senior or trusted members may have access to sensitive information, such as `resp_RFP` and `Bid_RFP` documents.

Of the above policy rules, R_1 and R_2 could be enforced by providing support for group-based mechanisms, whereas R_3 could be enforced by employing RBAC mechanism and R_4 that requires assignment of sensitivity levels to resources and trust levels to users, could be enforced by MAC mechanisms. Note, in particular, that an RBAC mechanism (by itself) can not provide support for limiting access to resources based on sensitivity and trust levels of resources and users respectively. In other words, no single access control model is sufficiently able to enforce all of the above requirements. Hence, we need to use a combination of access control models for supporting all of the above policy rules.

Recently, in order to incorporate requirements of hybrid access control policies into system design models, several visual modeling languages have been proposed, which are *directly* built on top of specific access control models [2, 7, 9, 12, 13, 17]. We note that, in order to provide support for new access control requirements, recent modeling languages of access control *either* propose *ad-hoc* extensions, which are appropriate for their requirements, to their preceding modeling languages *or* simply consider a combination of access control models as a basis for developing their modeling language. A "generic" modeling language of access control, from which a variety of access control models could *naturally* be derived and combined, would be a

more useful and simpler approach than existing approaches. However, such a generic modeling language requires a metamodel of access control as its basis.

3 Modeling Access Control Policies

Before proceeding further, we clarify the usage of terms *model* and *metamodel* in this paper. In particular, in the security community, an access control model describes a language for specifying access control policies. However, in the model community, such a language is called a *metamodel*. Hence, the term *model* in the access control community is analogous to the term *metamodel* in the model-driven community.

This means that, in modeling languages, an "access control model" is represented as a *metamodel* and a "metamodel of access control" is represented as a *meta-metamodel*. Since our work proposes a modeling language for access control and is closer to model-driven approaches, we chose to use the terminology of the model community.

In this section, we first describe our meta-metamodel and subsequently explain how specific metamodels are derived by instantiating our meta-metamodel.

3.1 Meta-metamodel

As mentioned earlier, the syntax of our modeling language is based on an *extension* of Barker's metamodel of access control [1]. Figure 1 illustrates our meta-metamodel. Essentially, the semantics our language mean that every association edge depicted between classes defines a relation on the classes.

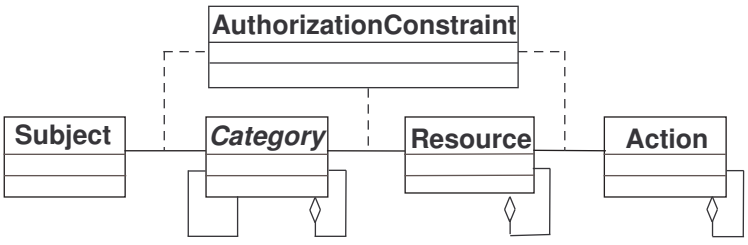


Figure 1: A visual representation of our meta-metamodel

The central component of our modeling language is the notion of a *category* that is instrumental to abstract key components of various access control models, such as *groups*, *security levels* and *roles*. Essentially, categories provide means to associate subjects and permissions, which are represented as `<resource,action>` pairs. Note that the class `Category` is defined as an *abstract* class (represented by italicized font). Hence, the class `Category` must be specialized for creating specific categories, such as groups, roles and security levels, to which subjects or resources can be assigned.

The left-side of class `Category` shows a class `Subject` that represents entities within the system that are able to initiate access request(s). The class `Subject` has an association with class `Category`.

The right-side of class `Category` shows two classes: `Resource` and `Action`. The class `Resource` represents protected resources within the system and is associated with the class `Category`. Such an assignment is useful to represent which resources are accessible by a given category.

The class `Action` represents operations that can be performed on protected resources. The class `Resource` has an association with class `Action`; thus, representing which actions can be performed on given resources.

It is sometimes desirable, for certain applications, to combine different access control models within a single application for supporting hybrid access control policies. Our modeling language provides support for hybrid access control policies by allowing categories to be associated with other categories (represented, in Figure 1, as a self-association edge on class `Category`). Furthermore, our language also allows to specify hierarchical relationships between categories by aggregating appropriate categories (represented, in Figure 1, as a self-association edge with a diamond head on class `Category`).

Note that our meta-metamodel also provides support for creating *resource hierarchy* and *action hierarchy* by aggregating resources and actions, respectively. Such hierarchical relationships are represented by an aggregation association (depicted in Figure 1 association edge with a diamond head) on classes `Object` and `Action`. Finally, the class `AuthorizationConstraint` is used to represent constraints that specify restrictions on subject-category assignments, category-resource assignments and resource-action assignments.

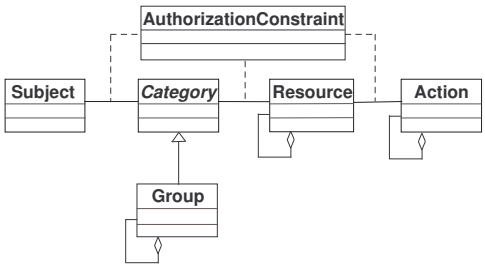
3.2 Deriving metamodels

We derive metamodels, which represent access control models, by instantiating the meta-metamodel. Recall that the class `Category` is defined as an

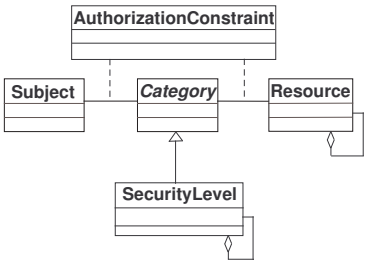
abstract class can be associated with other categories. Hence, by specializing the class `Category`, we can derive several different metamodels each of which represents group-based, mandatory or role-based access control, and even a combination of such metamodels for representing hybrid access control policies.

Figure 2 illustrates various ways to specialize the abstract class `Category`. Specifically, Figures 2(a), (b) and (c) illustrate the specialization of abstract class `Category` into classes `Group`, `SecurityLabel` and `Role`, respectively. Figure 2(d) illustrates the metamodel of a hybrid access control that associates specialized categories `Group`, `Role` and `SecurityLabel` with each other.

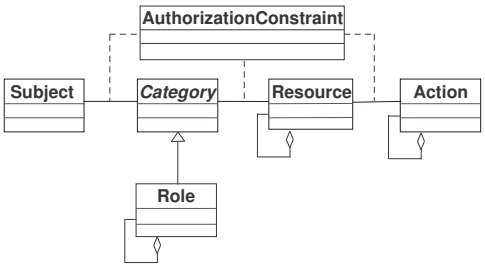
The "contains" relationship between categories is useful for modeling hierarchies of specialized categories; groups, security levels and roles, for example. In practice, however, it is unlikely that an instance of a specialized category contains instance(s) of other specialized categories.



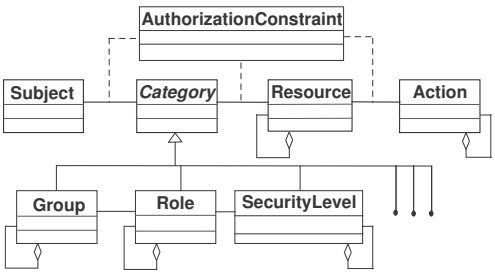
(a) Group-based access control



(b) Mandatory access control



(c) Role-based access control

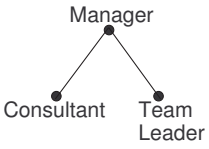


(d) Hybrid access control

Figure 2: Examples of metamodels by *specializing* class `Category`

3.3 Generating models

In this section, we describe the modeling of access control policies by instantiating the metamodels described in the previous section. Consider, for example, Figure 3 that illustrates an authorization state for the "bid submission" example introduced in Section 2. Figures 3(a), (b) and (c) show a role hierarchy, permission-role assignments and user-role assignments, respectively. Figure 3(d) illustrates a group hierarchy and Figure 3(e) illustrates a



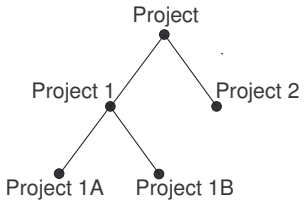
(a) Role hierarchy

Permission	Role
(Write,Input_RFP)	Consultant
(Read,Input_RFP)	Team Leader
(Write,Resp_RFP)	Team Leader
(Read,Resp_RFP)	Team Leader
(Write,Bid_Prop)	Manager
(Read,Bid_Prop)	Manager

(b) Permission-role assignment

User	Role
Alice	Consultant
Bob	Consultant
Carol	Consultant
Dave	Team Leader
Eve	Team Leader
Flora	Manager

(c) User-role assignment



(d) Group hierarchy

User	Group
Alice	Project 1A
Bob	Project 1B
Carol	Project 2
Dave	Project 1
Eve	Project 2
Flora	Project 1
Flora	Project 2

(e) User-group assignment

Figure 3: An example of RBAC configuration

user-group assignment relation that specifies groups to which users belong.

Figure 4 illustrates a concrete model that respects the metamodel of RBAC (shown in Figure 2(c)) for authorizations given in Figures 3(a), (b) and (c). Specifically, the left half of Figure 4 shows three roles `Consultant`, `Team Leader` and `Manager` and their assigned users, whereas the right half shows the assignment of resources to roles and actions that can be performed on resources. Note that our language denotes the association of actions to resources by a *stereotype* `<<AccessibleRole>>`, which specifies the set of roles to which the `<resource-action>` association applies. For example, in Figure 4, the association between resource `Input_RFP` and action `Write` is

specified for role `Consultant`, whereas the association of action `Read` to resource `Input_RFP` is specified for role `Team Leader`. Figure 4 also shows the hierarchical relation between roles, where a role towards the diamond end inherits the role on the other end of the (aggregation) association.

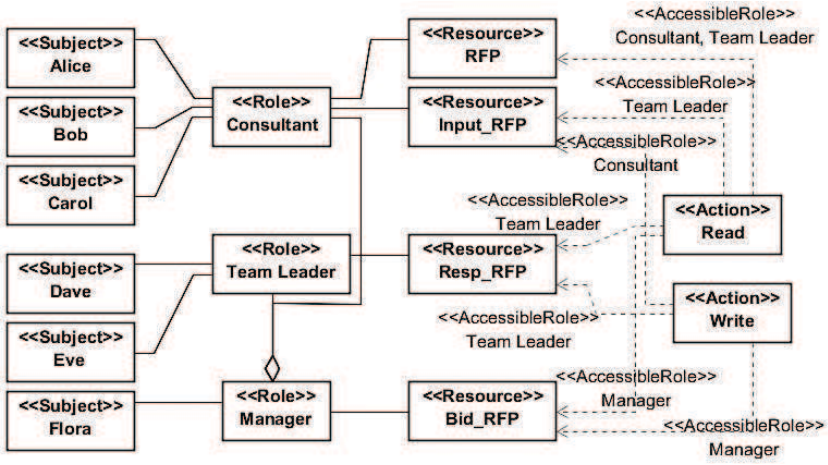


Figure 4: A model based on RBAC

Figure 5 extends Figure 4 by supporting *user groups* and illustrates the concrete model for the authorizations given in Figure 3. The concrete model of Figure 5 respects the metamodel of hybrid access control shown in Figure 2(d). The *user-group* assignments are shown by defining an association between a user and the groups to which the user belongs. For example, users *Alice* and *Bob*, who are assigned to role `Consultant`, belong to groups `Project 1A` and `Project 1B` respectively. Figure 5 also depicts the hierarchical relation between groups, where a group towards the diamond end inherits the group on the other end of the (aggregation) association. This means that user *Carol* who has membership of group `Project 1`, belongs to both groups `Project 1A` and `Project 1B`. Hence, user *Carol* can assume the role of a `Consultant` for *both* `Project 1A` and `Project 1B`; whereas user *Alice* can assume the role of `Consultant` *only* for `Project 1A`, for example.

Figure 6 extends Figure 5 by defining *security levels* of both users and resources for providing support for MAC policies. For example, resource `RFP` (request for proposal document) has a security (classification) level `Unclassified`. Hence, all users are permitted to `Read`, but *can not* write/modify, resource `RFP`. In contrast, resource `Input_RFP` has a security

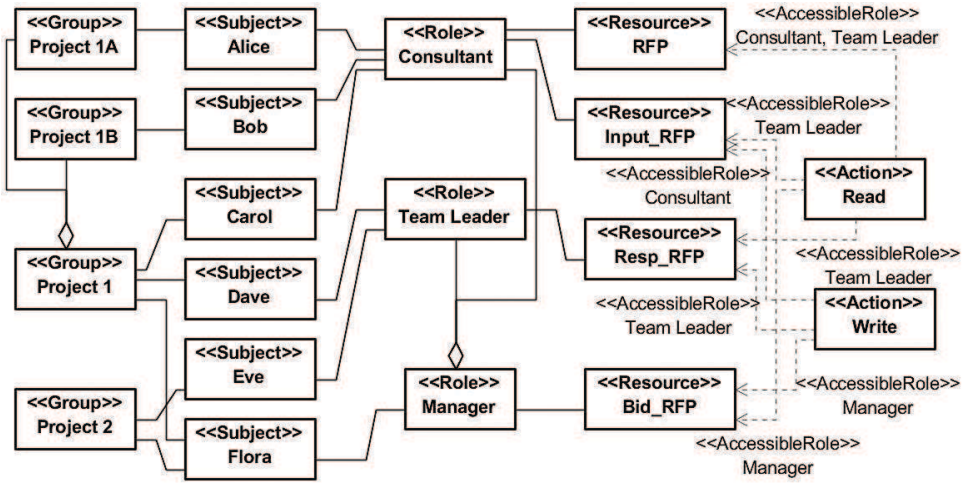


Figure 5: A model based on user-groups and RBAC

level `Classified`. Hence, users *Alice* and *Bob* who have a security level (`Unclassified`) that is lower than the security level of resource `Input_RFP` can only write to `Input_RFP`, but cannot read from `Input_RFP`. Whereas user *Carol* who has a security level that equals the security level of resource `Input_RFP` can both write to and read from resource `Input_RFP`.

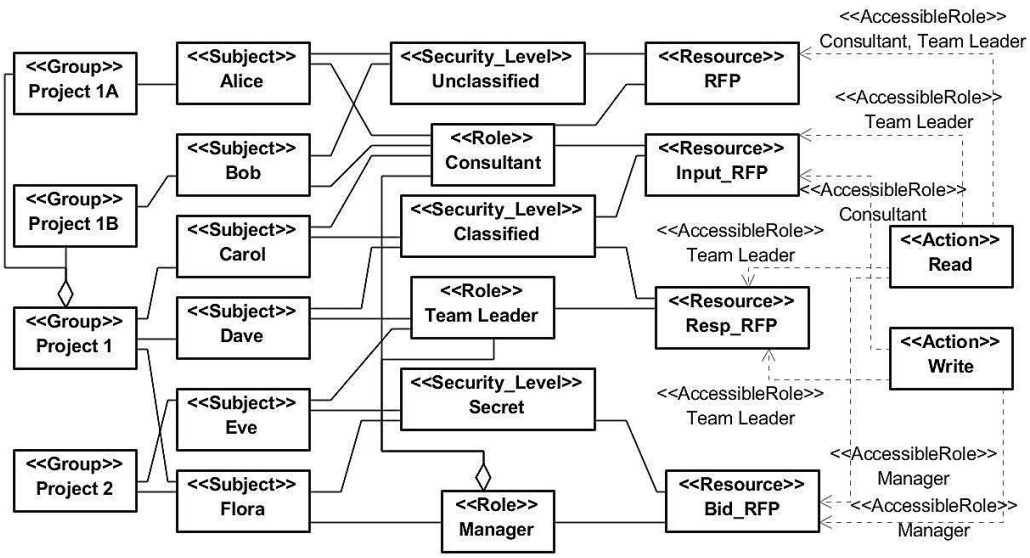


Figure 6: A hybrid model

We now describe the specification of authorization constraints in our modeling language. Recall *Rule 1* for the bid submission example from Section 2 that requires: only members of the concerned project must be able to access document `Input_RFP`.

Figure 7 illustrates the modeling of authorization constraint by using an OCL expression [10]. Essentially, the authorization constraint $subject.group.name- > intersection(resource.group.name)- > notEmpty()$ specifies that, in order to access resource `Input_RFP`, a subject (user) must have membership in a group to which resource `Input_RFP` belongs.

Assume, for example, that the resource `Input_RFP` is assigned to group `Project 1A` (shown in Figure 7 by an association between resource `Input_RFP` and group `Project 1A`). Then, user *Alice* is authorized to access resource `Input_RFP` because *Alice* has a membership in group `Project 1A` to which resource `Input_RFP` belongs. Whereas user *Bob*, who belongs only to group `Project 1B`, is not authorized to access resource `Input_RFP`.

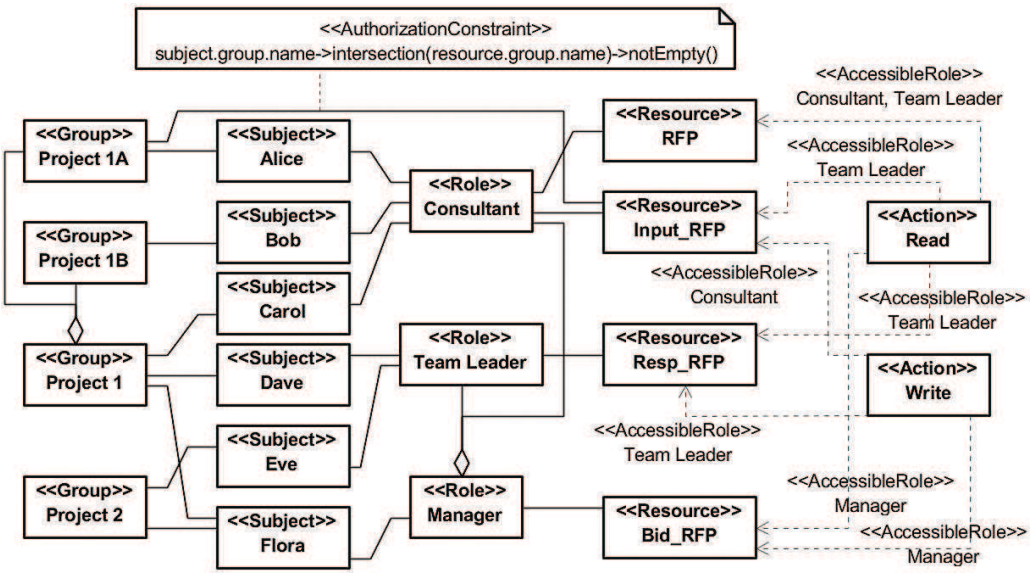


Figure 7: A hybrid model with authorization constraint

4 Related Work

Recent research has resulted in the development of UML-based modeling languages that incorporate security requirements into system design mod-

els [2, 7, 9, 12, 13, 17].

Epstein and Sandhu proposed a framework that uses UML notations for employing UML as a language to represent RBAC requirements [7]. Shin and Ahn proposed an alternative technique to utilize UML notation to describe RBAC modeling [17]. The work of Doan *et al* provided support for incorporating MAC into UML diagrams [6].

Jürjens proposed an approach, called *UMLsec*, for developing secure systems using an extension of UML [9]. UMLsec offers support for the annotation of UML models with formally specified security requirements, like confidentiality or secure information flow.

The seminal work of Basin *et al* demonstrated the application of MDS to the domain of access control [2]. This work includes a security modeling language, called *SecureUML*, that provides support for developing system design models which include access control requirements. SecureUML is developed on an extension of RBAC [2]. We believe that this feature of SecureUML limits its applicability when a diverse set of access control requirements are to be supported. For example, SecureUML *can not* express both non-RBAC and hybrid access control policies. In comparison, our modeling language is developed by using the concept of *category*, which can be specialized for creating various access control types, such as roles, security levels and groups. Hence, we believe that our modeling language is more generic than SecureUML; and regard SecureUML as a special case of our modeling language.

The work of Pavlich-Mariscal *et al* [12] is closer to our work. This work proposes a modeling language that provides support for a variety of access control requirements, such as DAC, MAC and RBAC, into security design models. Their modeling language also provides support for hybrid access control policies, which combine different access control models for specifying policy requirements.[2]

However, the modeling language is developed "directly" by using DAC, MAC and RBAC components. Hence, their modeling language can not support policies, other than DAC, MAC and RBAC. In contrast, our modeling language that is based on a generic concept of *category*, which can be specialized (as required) for creating specific access control types. This feature of our modeling language simplifies its syntax and provides a more generic approach for supporting different access control models.

[2]Note that what is called as a 'hybrid access control policy' in our work is referred to as *custom access control policy* in [12].

Furthermore, their modeling language *only* provides support for user-role assignment constraints, but *not* other types of constraints; permission-role constraints, for example. However, our modeling language supports the specification of constraints on user-category assignments, category-resource assignments and resource-action assignments.

Ray *et al* proposed parameterized UML elements for modeling an access control framework that combines MAC and RBAC in order to express hybrid access control policies [13]. This work provides support for modeling roles and security (clearance and classification) levels for subjects and resources. However, the UML modeling framework of their work is developed directly using MAC and RBAC models, whereas our framework is developed on the basis of a metamodel of access control.

5 Conclusion and future work

We have proposed a UML-based modeling language for incorporating access control requirements into system design models. The proposed modeling language is based on a metamodel of access control, rather than specific access control models. We have also extended Barker's metamodel of access control for providing support for *object hierarchies* and *action hierarchies*. We believe that this is the first work in the literature to develop a UML-based modeling language that is based on a metamodel of access control.

Adopting a metamodel of access control as a basis for developing our modeling language has been useful to develop a modeling language that is simple, yet sufficiently powerful to capture different access control models (both independently and combined). Specifically, our modeling language could be used as a generic framework for integrating a variety of access control requirements into system design models. We demonstrated that the meta-metamodel of our framework could be "specialized" into various metamodels, each of which represents a specific access control model.

Our immediate future work would be to develop techniques for verifying that the resultant models are consistent with their corresponding metamodels. Further, we intend to extend the proposed framework to provide support for attribute-based access control (ABAC) requirements [18]. Such requirements would be useful for specifying fine-grained and/or customized access control policies. For example, users who are assigned to the same set of roles may be required to be authorized for different sets of permissions, based on

the attributes defined in users' professional profiles.

We also aim to develop a tool for implementing the proposed framework and conducting a case study by using UACML (Unified Access Control Modeling Language).

References

[1] S. Barker. The next 700 access control models or a unifying meta-model? In *Proceedings of 14th ACM Symposium on Access Control Models and Technologies (SACMAT'09)*, pages 187–196, 2009.

[2] D. Basin, J. Doser, and T. Lodderstedt. Model driven security: From UML models to access control infrastructures. *ACM Transactions on Software Engineering and Methodology*, 15(1):39–91, 2006.

[3] D.E. Bell and L. LaPadula. Secure computer systems: A mathematical model. Technical Report MTR-2547, Volume I & II, Mitre Corporation, Bedford, Massachusetts, 1973.

[4] International Business Machines Corporation. IBM WebSphere Portal version 6.0, Administration. Available at `http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/topic/com.ibm.wp.ent.doc/wp_admin_pdf.pdf`.

[5] R. Crook, D. Ince, and B. Nuseibeh. On modelling access policies: Relating roles to their organisational context. In *Proceedings of 13th IEEE International Requirements Engineering Conference*, pages 157–166, 2005.

[6] T. Doan, S. Demurjian, T.C. Ting, and A. Ketterl. MAC and UML for secure software design. In *Proceedings of 2004 ACM workshop on Formal methods in security engineering (FMSE'04)*, pages 75–85, 2004.

[7] P. Epstein and R. Sandhu. Towards a UML based approach to role engineering. In *Proceedings of 4th ACM workshop on Role-based Access Control (RBAC'99)*, pages 135–143, 1999.

[8] D. Frankel. *Model Driven Architecture: Applying MDA to Enterprise Computing*. John Wiley & Sons, 2003.

[9] J. Jürjens. Towards development of secure systems using UMLsec. In *Proceedings of 4th International Conference on Fundamental Approaches to Software Engineering (FASE/ETAPS'01)*, volume 2029 of *LNCS*, pages 187–200, 2001.

[10] Object Management Group. Object constraint language, version 2.2, 2010. OMG Document Number: formal/2010-02-01.

[11] Object Management Group. OMG Unified Modeling Language, version 2.3, 2010. OMG Document Number: formal/2010-05-03.

[12] J. Pavlich-Mariscal, S. Demurjian, and L. Michel. A framework of composable access control features: Preserving separation of access control concerns from models to code. *Computers & Security*, 29(3):350–379, 2010.

[13] I. Ray, N. Li, D. Kim, and R. France. Using parameterized UML to specify and compose access control models. In *Proceedings of 6th IFIP WG 11.5 Conference on Integrity and Control in Information Systems (IICIS)*, 2003.

[14] Pierangela Samarati and Sabrina De Capitani di Vimercati. Access control: Policies, models, and mechanisms. In *Foundations of Security Ananlysis and Design*, volume 2172 of *LNCS*, pages 137–196. Springer-Verlag, 2001.

[15] R. Sandhu, E.J. Coyne, H. Feinstein, and C.E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[16] R. Sandhu and P. Samarati. Access control: Principles and practice. *IEEE Communications*, 32(9):40–48, 1994.

[17] M. Shin and G. Ahn. UML-based representation of role-based access control. In *Proceedings of 9th IEEE International Workshops on Enabling Technologies (WETICE'00)*, pages 195–200, 2000.

[18] E. Yuan and J. Tong. Attributed based access control (ABAC) for web services. In *Proceedings of IEEE International Conference on Web Services (ICWS'05)*, pages 561–569, 2005.