ACCD - Access Control Class Diagram

Martin Otto Werner Wagner

Department of Computer Science
Technische Universitaet Muenchen
Bolzmannstr. 3, 85748 Garching, Germany
martin.wagner@tum.de

Abstract: The access control logics of an information system may become large and complex. To develop a system that fulfils the customer's requirements regarding access control three conditions are to be satisfied. These requirements are to be complete, unambiguous and defined by the customer. These three objectives are conflicting. Freely defined requirements, formulated in the customer's native language, may be incomplete and ambiguous. More formalised requirements, for example in an XML format, can be complete and unambiguous but may not be formulated or even understood by the customer. This work shows a diagram type based on the UML class diagram that helps to define complete and unambiguous access control logics. It aims to be used by non-IT experts like UML itself.

1 Introduction

Defining the access control of an IT system is basically just answering the question "who can do what?". The answer to this question is not as simple as it might appear at first. For instance it is not obvious that parts of the access control logic are realised in the system itself (static) others are to be configurated in this system at runtime (dynamic).
Sometimes there is a choice whether an aspect of the access control logic is dynamic or static. It is conceivable that based on the same access control requirements roles could be implemented statically and dynamically. Roles are here understood as the concept of Role Based Access Control (RBAC) [FK92] describes. Of course the usage and usability of theses two different implementations might differ. In this example the dynamic definition of roles would lead to more (non-functional) features but most likely also to more costs. Consequently a reasonable distinction between static and dynamic access control as well as a refined definition of the static access control is important. It is crucial for the correct implementation of the functional requirements, the usability and costs of the system.
The person who defines the requirements of a system (hereinafter referred to as customer) is often no IT expert but an expert of the problem domain. Nevertheless the customer has to be aware of the consequences of his decisions. In contrast to costs the consequences for security and usability cannot be quantified. Thus the customer has to understand the differences in detail, even though he is no IT expert. Two combinable approaches can help the customer during that process. Firstly IT experts can assist the customer, secondly a

comprehensible notation that covers the aspects access control may be applied.
There is no comprehensible way to describe the access control logic of IT systems. For the general description of a system UML is relatively comprehensive [RSP07]. However, UML has no dedicated concept for modelling access control. A combination of class and use case diagrams can be utilised to define simple access control logic. This approach is not sufficient because the models become enormous and certain requirements can't be modelled. In other approaches UML-Profiles define a set of UML-extensions for class diagrams. None of the approaches support the description of the entire access control logic in a comprehensible way. Their shortcomings are for example: class diagram elements are multiplied by a factor of more than two, cryptic description of logics, class elements and their access control logic is described separately, access control logic and program logic is mixed together, and the modelling of important access control aspects is not possible.
This idea paper is structured as follows: In chapter 2 important concepts of access control, that should be expressible in modelling languages for access control, are defined. Chapter 3 gives an overview of the existing relevant approaches and explains their capabilities, strengths and weaknesses. In chapter 4 an approach, based on UML class diagrams, will be introduced and explained. Some ideas for future work are described in chapter 5.

2 Important concepts for access control logic

Firstly a class diagram of an application for managing calendars is shown (figure 1) and some exemplary requirements are stated. On the basis of this class diagram important concepts for access control logic are stated. A modelling language for access control should offer to express these concepts.

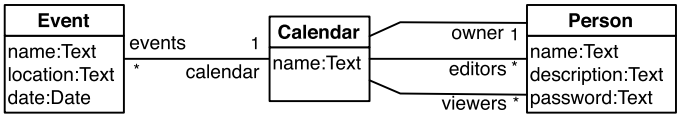

Figure 1: Example: class diagram of a simple calendar application

Exemplary (access control) requirements: The application manages calendars that can have several Events and Persons. Everybody may register to the application as a Person, each registered Person may create a new calendar. Different Persons can access a calendar with different rights which are described in roles (Owner, Editor, Viewer). A Person who has a role on a calendar has automatically the same role on all associated Events and Persons. A Viewer of a calendar may read name of the calendar, read all attributes of the calendars events, and read name and description of all viewers and editors and the owner of the calendar. A Editor of a calendar has all Viewer rights and can add Events and change attributes of Events and may add and remove Persons as Viewers. An Owner of a calendar has all Editor rights and can remove Events and the calendar itself, may add and remove Persons as Editors and change the Owner to another Person. Each Person has the role Self on itself that allows to read and change all attributes of it.

Important concepts: A role based assignment of rights is important as the single rights

for a calendar with its associated events and persons should not be defined for each editor separately (**roles**). It has to be defined who owns which role (**role assignment**). A hierarchical approach like described in RBAC is desirable (**role hierarchy**) as it simplifies the definition of the roles editor and owner. Rights can differ between the instances of the same class. This is required as different calendars can have different owners (**instance distinction**). Rights can differ between the properties of an object as a third party may see the name but not the password of a Person (**property distinction**). Some rights have to be granted without an existing object. For example a calendar should be creatable without its previous existence (**static rights**). Other rights have to be granted without an existing subject. For example someone should be able to register to the system without his previous existence in the system (**anonymous rights**). Rights on an object can include rights on other objects. For example a calendars owner should be automatically the owner of the calendars events (**transitive inclusion**). Subjects are accessible objects as the owner of a calender should bee seen (**subject access**). The bold written requirements names will be used in chapter 3 for classifying the existing approaches.

3 Related work and an overview of access control model types

The Unified Modeling Language (UML)[BRJ96] is a common [DP06] visual modelling language. As an extension to UML the Object Constraint Language (OCL) [Rat97, RG98] has been defined for formal definitions on models. It allows a modeller to specify the characteristic of an instance of a model. Role-Based Access Control (RBAC)[FK92, SCFY96] is a common[OL10] standard[FSG$^+$01] to manage access rights in IT systems. In the first column the names of the different approaches are stated. In the following columns the concepts defined in chapter 2 are evaluated, $\surd$ tells that this concept is expressible. In the last column statement for the comprehensibility is given (1 = bad, 2 = medium, 3 = good) which reflects the personal opinion of the articles author.

approach	roles	role assignement	role hierarchy	instance distinction	property distinction	static rights	anonymous rights	transitive inclusion	subject access	comprehensibility
Epstein/Sandhu	$\surd$	$\surd$	$\surd$		$\surd$	$\surd$	$\surd$		$\surd$	2
UMLsec					$\surd$	$\surd$	$\surd$		$\surd$	2
Shin/Gail-Joon	$\surd$				$\surd$	$\surd$	$\surd$		$\surd$	2
SecureUML	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$		$\surd$	1
Kuhlmann/Sohr/Gogolla	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$		$\surd$	1
ACCD	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	

Figure 2: Overview of the expressiveness of access control models

In the following the existing graphical approaches are shown for modelling access control.

Some of them have been named by their creators, the others are named by their authors.

Epstein/Sandhu This graphical modelling approach has a focus of expressing the concept of RBAC [ES99] for IT systems. Is is designed for a specific software development tool, the Role Based Access Control Framework for Network [TOB98], thus its range of application is very reduced. **UMLsec** UMLsec[Jür02, BJN07] is an UML based approach for modelling security aspects that is much broader than the aim of defining access control. It focuses more the communication and encryption and access control rights can't be defined very specifically. **Shin/Gail-Joon** An OCL based approach [SGJ00, AS01] defines access control logics via constraints (e. g. separation of duty constraints). While it is very powerful in defining permissions it doesn't focus on roles and role assignment. **SecureUML** is a modelling language on the basis of UML and OCL is SecureUML [LBD02]. SecureUML provides the broad spectrum of constraints as it is based on OCL. **Kuhlmann/-Sohr/Gogolla** [SMBA08, KSG13] uses UML and OCL to define access control logic that bases concepts of SecureUML. It separates the access control logic from the application logic. This allows subsequent development of access control functionalities without the necessity of change the application itself.

4 ACCD: The Access Control Class Diagrams

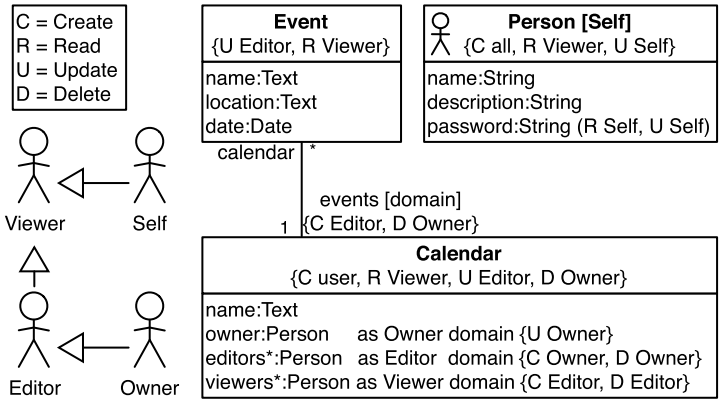

Figure 3: The Access Control Class Diagram for the calendar application

In this chapter the calendar application is modelled in an Access Control Class Diagram. On the basis of this model Access Control Class Diagrams are explained. Access Control Class Diagrams are based on class diagrams. The class diagrams are enriched by the information that is necessary to define access control logic. All important concepts (defined in chapter 2) can be expressed by this approach. **Roles** can be expressed with the actor symbol of use case diagrams. A **role hierarchy** can be defined by using arrows between the defined roles. A role can include another roles rights by pointing with an arrow on it. **Role assignment** can be defined with the keyword **as**. The **as** can only be used in references to subjects. The referenced subjects will gain the Role which is stated after the **as**. **Instance**

distinction is achieved as the referenced subjects are defined in the concrete instance individually. **Property distinction** is achieved as the rights can be defined in the granularity of single properties. If not rights are defined for a property the rights defined for the class will be inherited. On a class the rights create, read, update and delete can be defined while create allows to create a instance of the class and delete allows to delete this concrete instance of the class. The rights read and update are only for the inheritance to attributes of the class. Attributes are the properties of a class on which the defined type behind the colon are a primitive data type of the set $(Int, Double, Float, String, Boolean, Date)$. For an attribute the right read allows to see its value and update allows to change its value. References are properties of a class on which the defined type behind the colon is the Name of an existing class or subject class. Subjects are classes which are tagged with a stick-figure. On references the right create allows to add a referenced instance and the right delete allows to remove a referenced instance. Read and update rights on referenced instances can be defined via transitive inclusion. **Static rights** can be defined with the keyword **user**. Any registered user of the system can execute the rights assigned to **user**. **Anonymous rights** can be granted with the keyword **any**, these rights can be performed without being a subject in the system. **Transitive inclusion** is defined with the keyword **domain**. The roles assigned to an instance are automatically applied to the instances which are referenced with keyword **domain**. **Subject access** is allowed as subject classes can be referenced like normal classes.

5 Future Work

There are many open topics for future work. The evaluation of existing approaches should be extended. The concepts which are important for access control should be evaluated. Either through inquiring software developers or the usage of the ACCD in real projects. For the usage of ACCD in real projects two aspects are interesting: Is all required access control logic definable in ACCD and do customers and software developers understand the concept of ACCD. The observation of customers and software developers during the definition of access control could show the problems they usually face. A transformation logic can be defined which allows the usage of ACCDs as the model of model driven development, code generation directly from an ACCD is desirable.

References

[AS01] Gail-Joon Ahn and Michael E Shin. Role-based authorization constraints specification using object constraint language. *Proceedings Tenth IEEE International Workshop on Enabling Technologies*, pages 157–162, 2001.

[BJN07] Bastian Best, Jan Jurjens, and Bashar Nuseibeh. Model-based security engineering of distributed information systems using UMLsec. In *Software Engineering, 2007. ICSE 2007. 29th International Conference on*, pages 581–590. IEEE, 2007.

[BRJ96] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The unified modeling language*. University Video Communications and the Association for Computing Machinery, 1996.

[DP06] Brian Dobing and Jeffrey Parsons. How UML is used. *Communications of the ACM*, 49(5):109–113, 2006.

[ES99] Pete Epstein and Ravi Sandhu. Towards a UML based approach to role engineering. In *Proceedings of the fourth ACM workshop on Role-based access control*, RBAC '99, pages 135–143, New York, NY, USA, 1999. ACM.

[FK92] David F Ferraiolo and D Richard Kuhn. Role-Based Access Controls. *National Computer Security Conference*, pages 554–563, January 1992.

[FSG+01] David F Ferraiolo, Ravi Sandhu, Serban Gavrila, D Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, 2001.

[Jür02] Jan Jürjens. UMLsec: Extending UML for Secure Systems Development. In *UML '02: Proceedings of the 5th International Conference on The Unified Modeling Language*. Springer-Verlag, September 2002.

[KSG13] Mirco Kuhlmann, Karsten Sohr, and Martin Gogolla. Employing UML and OCL for designing and analysing role-based access control. *Mathematical Structures in Computer Science*, pages 796–833, 8 2013.

[LBD02] Torsten Lodderstedt, David A. Basin, and Jürgen Doser. SecureUML: A UML-Based Modeling Language for Model-Driven Security. In *Proceedings of the 5th International Conference on the UML*, UML '02, pages 426–441. Springer-Verlag, 2002.

[OL10] Alan C. O'Connor and Ross J. Loomis. 2010 Economic Analysis of Role-Based Access Control. *RTI International report for NIST*, December 2010.

[Rat97] Rational Software Corporation. Object Constraint Language Specification, 1997.

[RG98] Mark Richters and MARTIN GOGOLLA. On formalizing the UML object constraint language OCL. *Conceptual Modeling–ERâ98*, pages 449–464, 1998.

[RSP07] Rozilawati Razali, Colin F Snook, and Michael R Poppleton. Comprehensibility of UML-based formal model. In *WEASELTech '07*. ACM Request Permissions, November 2007.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-Based Access Control Models. *Computer*, 29(2):38–47, February 1996.

[SGJ00] M E Shin and Gail-Joon. UML-based representation of role-based access control. *IEEE 9th International Workshops on Enabling Technologies Infrastructure for Collaborative Enterprises*, pages 195–200, 2000.

[SMBA08] Karsten Sohr, Tanveer Mustafa, Xinyu Bao, and Gail-Joon Ahn. Enforcing role-based access control policies in web services with UML and OCL. In *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, pages 257–266. IEEE, 2008.

[TOB98] Dan Thomsen, Dick OâBrien, and Jessica Bogle. Role Based Access Control Framework for Network Enterprises. *Proceedings of 14* Annual Computer Security Application Conference*, pages 50–58, dec 1998.