A PATTERN SYSTEM FOR ACCESS CONTROL

Torsten Priebe[1], Eduardo B. Fernandez[2], Jens I. Mehlau[3], Günther Pernul[1]
[1]Department of Information Systems, University of Regensburg, D-93040 Regensburg, Germany; [2]Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431, USA; [3]Bavarian Research Cooperation for Information Systems (FORWIN), University of Regensburg, D-93040 Regensburg, Germany

Abstract: In order to develop trustworthy information systems, security aspects should be considered from the early project stages. This is particularly true for authorization and access control services, which decide which users can access which parts of the system and in what ways. Software patterns have been used with success to encapsulate best practices in software design. A good collection of patterns is an invaluable aid in designing new systems by inexperienced developers and is also useful to teach and understand difficult problems. Following in this direction, this paper presents a pattern system to describe authorization and access control models. First, we present a set of patterns that include a basic authorization pattern that is the basis for patterns for the well-established discretionary and role-based access control models. Metadata access control models have appeared recently to address the high flexibility requirements of open, heterogeneous systems, such as enterprise or e-commerce portals. These models are complex and we use the basic patterns to develop a set of patterns for metadata-based access control.

Key words: Access control models; authorization; computer security; object-oriented design; security patterns.

1. INTRODUCTION

Security plays an important role in any type of information system, especially when these systems are opened to the outside world, as in the case of web-based enterprise or e-commerce portals. However, security cannot be addressed in a consistent way when being integrated into the system after the system has been built. Security needs to be considered for the analysis and design stages and throughout the whole system development. This is particularly true for the authorization and access control components or services, which decide about what parts of the system specific users can access and in what ways. The entities involved in access control are subjects (active entities, users or processes trying to access system resources) and objects (passive entities, resources such as documents or system functions). Permissions (authorization rules) are used to define which subject can access which object and in which way. For this purpose, a number of access control models have been developed and discussed in the literature, with the Role-based Access Control model (RBAC) having been proposed as a standard [13; 14]. However, the standard RBAC model and its variations cannot consider the requirements imposed by open, heterogeneous systems, and new models that make use of metadata have appeared. We call these Metadata-based Access Control models (MBAC) and we present here patterns for this type of models.

Software patterns have been used with success to encapsulate best practices in software design. A good collection of patterns is an invaluable aid in designing new systems by inexperienced developers and is also useful to teach and understand complex systems such as this type of models. A special type of pattern, the security pattern has appeared recently and is becoming a hot topic for secure systems designers [19; 30]. This interest is shown by the publication of several new patterns, the appearance of a book [31], and recent pages for security patterns in the IBM and Microsoft web sites. In order to provide a structured presentation of the access control models and to support their use during the information system design, we use the concept of security patterns. A pattern system is being developed which identifies basic building blocks of access control linking them to the well known discretionary and role-based access control models as well as to the novel metadata-based model. A related approach considers access control in the context of specific applications, tries to detect policy conflicts, and attempts to find ways to compose different access control models [23; 29].

The remainder of this paper is structured as follows: In section 2 we review the concept of security patterns. These ideas are used in section 3 to identify building blocks for access control models. Based on these, we present a pattern for role-based access control in section 4. As said earlier, the limited flexibility properties of RBAC lead to the development of the Metadata-based Access Control model, which is described – also as a pattern – in section 5. Section 6 ends the paper with some conclusions and discusses possible future work.

2. SECURITY PATTERNS

The knowledge of experienced developers can be embodied in patterns, which describe proven solutions for recurring problems. In particular, patterns offer the possibility of collecting, systematizing, and cataloguing the know-how of many designers. Patterns thereby facilitate the generation and documentation of well-known and established solutions. These encapsulated solutions are very valuable when building new systems, evaluating existing systems, and can improve the communication and learning process [6, p. 5]. Patterns usually include a problem description within a specific context with a corresponding solution. "A Pattern is an idea that has been useful in one practical context and will probably be useful in others." [21]

A multiplicity of different kinds of patterns has been considered in the literature: patterns are developed for various levels of abstraction, ranging from fundamental paradigms for structuring software systems or even management patterns to concrete implementations of particular design decisions. Thus, patterns accompany not only one phase of the software development process, but give support from analysis to the implementation phase. For example analysis patterns support conceptual modeling [4; 21]. Design patterns refer to ways to improve flexibility and other characteristics of the software [22], while architectural patterns describe fundamental structuring principles of software systems on the basis of predefined subsystems and constraints for the organization of their relationships [6].

Patterns are described in natural language and usually enriched with semi-formal diagrams, typically UML diagrams. Patterns are described using templates that have a fixed structure. The two best-known pattern templates are those from Gamma et al. (GoF template) and Buschmann et al. (POSA template) [6; 22].

During the development of a software system many different patterns are used. Therefore, it is necessary to group the patterns in a consistent way. In particular the interdependences between the patterns have to be described. The most common ways are pattern systems, which are collections of related patterns and pattern languages, where there is an implication of completeness.

Isolated security patterns and first approaches for pattern systems for selected problem domains have been increasingly appearing in the literature. The security pattern approach was first used in 1997 for the description of general security measures for information systems [32], although [17] and [12] already had shown object-oriented models for security. Security patterns for the design of cryptographic software components [5] and for access control models followed [3; 16; 18; 24]. [30] describes a theoretical model for security patterns, which formalizes relationships between patterns. A coming book catalogs a variety of security patterns [31].

Similarly to the classical pattern approach, security patterns are represented in a structured form. The uniform description enables an easier comparison and a systematic searching for users. The description template used in the following supports these requirements and considers five fundamental elements in its structure. These sections are a common set of the two main templates and include: intent, context, problem, solution, and consequence section. The name of the security pattern extends the domain vocabulary and facilitates the discussion and documentation. Its intent describes succinctly the problem being solved by the pattern. The context describes the environment or situations where the pattern applies and is useful. The problem section describes a problem that needs an appropriate solution and which conditions have to be met in order to be able to use the pattern. The solution section describes a generic solution to a problem, indicating with class, sequence, and other UML diagrams, the form of the solution. Finally, the consequences section describes the effect of the pattern as well as constraints on its use. For security patterns a statement about its effect on the system security goals should be given.

In the following, security patterns for authorization and access control models are developed and presented in the form of a pattern system.

3. A PATTERN SYSTEM FOR ACCESS CONTROL

In the following we will present access control patterns that demonstrate the fundamental principles of authorization and access control models and serve as guidelines for secure systems design. First we identify and describe two building blocks for access control models, the Authorization Pattern and the Session Pattern. In the next section we will describe a pattern for the RBAC access control model as a specialization of these building blocks. We have also developed a pattern for the classic Discretionary Access Control (DAC) model [7; 26], also based on the Authorization Pattern, but due to space limitations cannot show now details here.[1]

Finally, two patterns for an access control model based on metadata (Metadata-based Access Control – MBAC) will be presented, the MBAC Pattern and the MBAC Pattern with Sessions as an extension of it. In addition, we define an MBAC Pattern with Predicates which borrows the idea of predicates (further restrictions of the access rights) from DAC and a Com-

[1] Of course, for the sake of completeness, the Mandatory Access Control model (MAC) [7; 26] could also be expressed as a pattern. However, it is not based on the idea of explicit authorizations and thus a MAC pattern would not be based on our Authorization Pattern.

posite MBAC Pattern, but again do not give now details due to space limitations. Figure 1 shows all these patterns and their interdependences as a pattern system in UML notation.

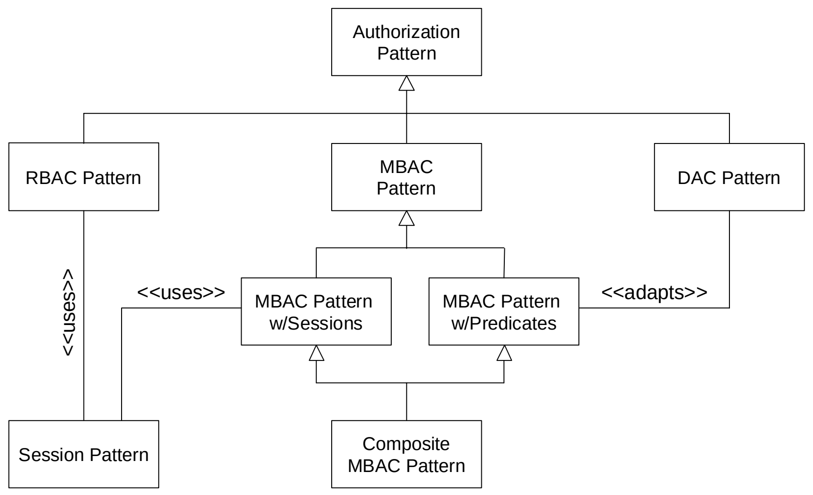

Figure 1. Pattern system for access control

3.1 Authorization Pattern

- **Pattern name:** Authorization Pattern
- **Intent:** How do we describe who is authorized to access specific resources in a system?
- **Context:** Any environment where we need to control access to computing resources.
- **Problem:** Granted permissions (authorizations) for security subjects accessing protected objects need to be indicated explicitly. Otherwise, any subject could access any resource. The model should be independent from the type of resource to be protected. Administration of authorizations should be supported in an efficient and consistent manner.
- **Solution:** Indicate in a suitable representation the concept of subject, object, and access type. Figure 2 shows the elements of an authorization in form of a class diagram. The class *Subject* describes an active entity, which attempts to access a resource (protected object) in some way. The class *Object* represents the resource to be protected. The association between the subject and the object is called *Authorization*, from where the pattern receives its name. The association class *Right* describes the access type (e.g. read, write) the subject is allowed to perform on the corresponding object.

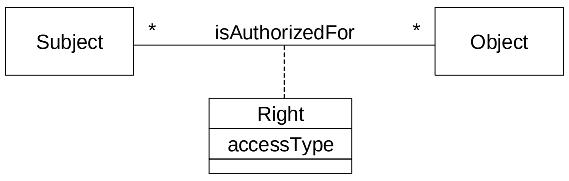

Figure 2. Authorization Pattern

- **Consequences:** The solution is independent from the resources to be protected. A resource can be a memory area, I/O device, file, or database table. Access types are individually definable and can be application specific (in addition to the usual read and write).
- **Known uses:** This pattern defines the most basic type of authorization rule, on which most other (more complex) access control models are based. Packet filtering firewalls implement a variety of this pattern where the subjects and objects are defined by Internet addresses.

3.2 Session Pattern

- **Pattern name:** Session Pattern
- **Intent:** Provide an environment where the rights of a user can be restricted and controlled.
- **Context:** Any environment where we need to control access to computing resources.
- **Problem:** Depending on the context, e.g. within a certain application, a user should activate only a subset of the authorizations that he has. This prevents the user from mistakenly misusing his rights (e.g. accidentally deleting certain files). Also, if an attacker compromises a running process his potential for damage is reduced.
- **Solution:** Define a context for the user interactions, where controls and logging can be applied. Figure 3 shows the elements of a session in form of a class diagram. The class *Subject* describes an active entity, which accesses the system and requests resources. The class *AuthorizationContext* describes the set of execution contexts or active rights that the user can exert in a given interaction. A subject can be in several sessions at the same time. A session has a limited lifetime. When starting a session (e.g. by logging in), a user activates only a subset of the authorization contexts (e.g. roles) assigned to him, so that only the necessary rights are available within this session. Note that we see the Session Pattern as a building block for access control models. Thus, we have an authorization (rather than authentication) centric view on sessions in this model.

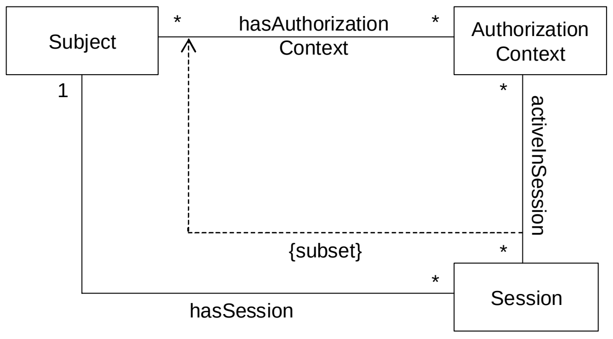

Figure 3. Session Pattern

- **Consequences:** Sessions permit the system to implement the principle of "least privilege". Every session gets just as many privileges as needed to perform the desired tasks. In this way the potentially harm is reduced when a session is compromised, because only the activated subset of the authorizations can be misused. Moreover, the concept protects subjects from (unintended) misuse because during a session they can use only operations allowed by the activated authorizations.
- **Known uses:** This concept appears in many computational environments, e.g. RBAC and a variation of MBAC use sessions as defined by this pattern.

4. ROLE-BASED ACCESS CONTROL PATTERN

- **Pattern name:** RBAC Pattern
- **Intent:** Control access to resources based only on the role of the subject.
- **Context:** Any environment where we need to control access to computing resources and where users can be classified according to their jobs or their tasks.
- **Problem:** Permissions for subjects accessing protected objects have to be described in a suitable way. A central authority should be responsible for granting the authorizations. Furthermore, a convenient administration of the authorizations should be guaranteed for a large number of subjects and objects. The principle of least privilege should be supported.
- **Solution:** The solution presented in the RBAC Pattern is based on the model of [16], adapted to represent the standard proposed in [13; 14][2].

[2] More precisely, this model corresponds to what is called "Hierarchical RBAC" in [13; 14]. Constraints, which could be used to implement static and dynamic separation of duty, are not considered at this point.

Figure 4 is based on the classes of the Authorization Pattern and the Session Pattern, i.e., this is a composite pattern. The class *Role* disconnects the direct link between subjects and objects and represents organizational roles and responsibilities, which can be nested hierarchically. Subjects can be assigned to several roles. Roles are given authorizations for different objects. Within a session a subject can activate a subset of the roles assigned to her, i.e. only those that are necessary to perform her current tasks.

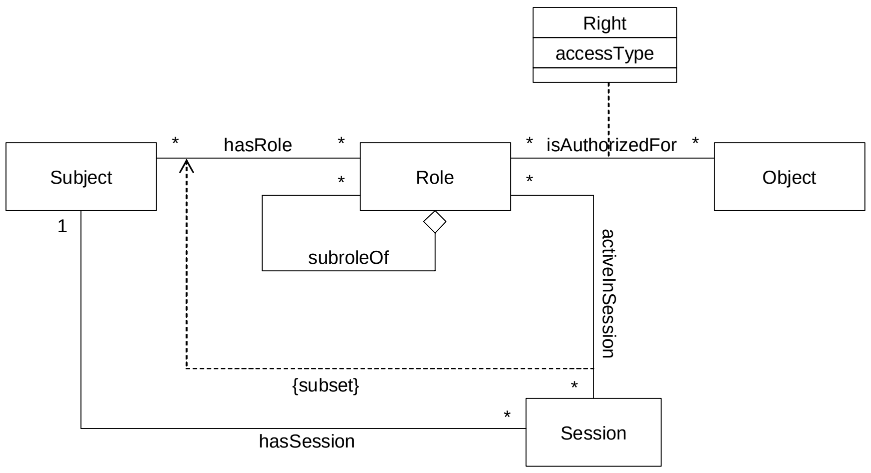

Figure 4. RBAC Pattern (adapted from [16])

- **Consequence:** By introducing roles the administrative effort is reduced, because there is no need to assign rights to individuals. The structuring of roles allows larger groups to be handled. By means of the session concept the implementation of the least privilege principle is supported.
- **Known uses:** The role-based access control model has been standardized by the NIST [13; 14]. It is also used in operating systems such as Solaris (for controlling administrative rights), and in the Oracle DBMS. Component frameworks such as J2EE and .NET also use roles to assign rights [20].

5. PATTERNS FOR METADATA-BASED ACCESS CONTROL

The Role-based Access Control model presented in the past section simplifies the administration of authorizations. However, for very large open systems such as digital libraries, enterprise or e government portals, or hospital systems, the role hierarchies can become very complex. One possibility to address this issue is the use of tools to help in administrating RBAC systems [11]. However, when the number of protected objects also increases, a manual assignment of authorizations becomes very expensive and error-prone. Furthermore, in many situations access depends on contents of an object and the environment the subject is acting in. In these applications we need to deal with users not previously registered. The Metadata-based Access Control model provides a more convenient and efficient way to manage access rights for these situations.

The basic idea is to utilize (possibly dynamic) properties of subjects and objects as the basis for authorization, rather than directly (and statically) defining access rights between users, roles, and objects. On the user side, an attribute could be his position within an organization, quite similar to a role. Especially for external users however, acquired credentials (e.g. subscriptions, customer status) or attributes such as age or shipping address may need to be used instead of predefined rights. For the security objects, the content, e.g. of documents, can be described by means of metadata. Such metadata elements should be used for authorization purposes.

Two primary directions of research have evolved. The first derives from research on security for digital libraries. [1; 15] propose a Digital Library Access Control Model (DLAM), which defines access rights according to properties associated with subjects and objects. Possible objects are text or multimedia documents, but also parts of them (e.g. individual paragraphs). DLAM supports positive authorizations (explicit access permissions) as well as negative authorizations (explicit denial of access).

The second important direction of research has its origin in the area of public key infrastructures (PKI) and is based on the use of certificates for authentication. A widespread standard for certificates is X.509 [25], which enables the user to employ his private key for authentication, while the respective addressee is using the certified corresponding public key for checking the claimed identity. In addition to the public key, also other attributes can be assigned to the owner of a certificate. [2] proposes to use these attributes for authorization and access control purposes.

In industry, Microsoft's .NET framework uses a model based on membership conditions, code groups, and policy levels which is used to evaluate access and is close to the models indicated above [26].

Based on these considerations a pattern for Metadata-based Access Control will be developed in the following. As enhancement and differentiation from DLAM the model will be extended by a session concept, as used in the RBAC model. The use of attribute certificates presents a further potential enhancement to the model and will be left for future work.

5.1 Metadata-based Access Control (MBAC) Pattern

- **Pattern name:** MBAC Pattern
- **Intent:** Control access based on properties of subjects or objects.
- **Context:** Any environment where we need to control access to computing resources and where some users may not be pre-registered.
- **Problem:** Similarly to the other patterns in section 4, permissions for subjects accessing security objects have to be described in a suitable way. The administration of authorizations should be simplified as done in Role-based Access Control. In addition, in open systems such as web portals we usually don't know the subjects in advance. Access may also be dependent on values of the object; for example, a patient can access her own record.
- **Solution:** Subjects in requests and actual requested objects are represented as sets of attribute or property values. In addition, we need to represent authorization subjects and objects as sets of predicates or assertions on attribute or property values, i.e. the authorizations are not defined directly between subjects and objects but between so called subject and object descriptors. A subject descriptor consists of several attribute conditions (e.g. age > 21, ZIP code beginning with "93") which can possibly correspond to several real subjects. The same holds for the object descriptors, where conditions are defined on object properties (e.g. related to a certain project, released by a certain publisher). As a consequence, subject and object descriptors are something like subject and object groups, however, not explicitly grouped by an administrator, but implicitly by their attribute or property values.

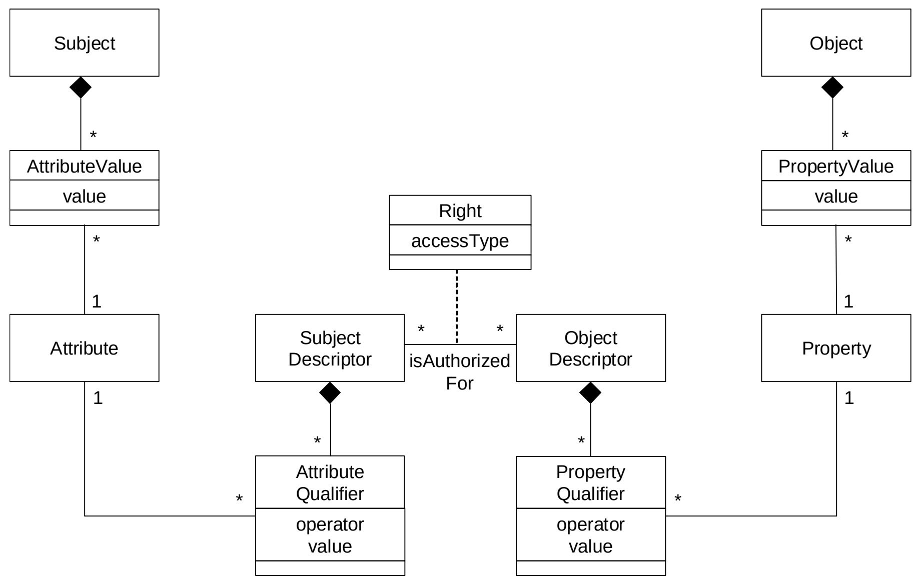

Figure 5. MBAC Pattern

Figure 5 shows the elements of the MBAC model in form of a class diagram. This is a composite pattern that uses the Authorization Pattern. The class *Subject* describes the actual accessing entity. A subject is described by several attributes values (as instances of the *AttributeValue* class). The class *Attribute* denotes the attribute schema (e.g. an instance of the class would be age). A similar decoupling has been done for the class *Object*. The object represents the resource which has to be protected and is described by property values, represented by the class *PropertyValue*. Rights are defined between subject and object descriptors (represented by the *SubjectDescriptor* and *ObjectDescriptor* classes). Like the attribute and property values the assertions that define the descriptors have been decoupled into the *AttributeQualifier* and *PropertyQualifier* classes.

- **Consequences:** By using subject attributes and object properties for the definition of authorizations, administration is simplified and flexibility is improved. When changing attribute values, affected permissions will be automatically updated without the need to explicitly change role definitions. Subjects can be roles as well as users or processes.
- **Known uses:** In addition to the research models mentioned above, this pattern is used in the authorization system for the .NET component framework [26].

5.2 MBAC Pattern with Sessions

- **Pattern name:** MBAC Pattern with Sessions
- **Intent:** Control access based on subject and object attributes within the context of a session.
- **Context:** Any environment where we need to control access to computing resources.
- **Problem:** Again, permissions for subjects accessing security objects have to be described in detail. The administration of authorizations shall be simplified and the requirements of open and heterogeneous systems such as portal systems have to be particularly considered. In addition, the principle of least privilege should be supported.
- **Solution:** The MBAC Pattern with Sessions shown in figure 6 is an extension of the MBAC Pattern described in the previous section, combined with the Session Pattern from section 3.2. As in the MBAC Pattern, subjects are described by some attributes and objects are described by some properties. The authorizations are defined between subject and object descriptors. For this MBAC extension the concept of sessions has been added. Similar to the RBAC model the user has the option to activate only a subset of the attributes assigned to him within a session. Only the activated attributes are used for access control.

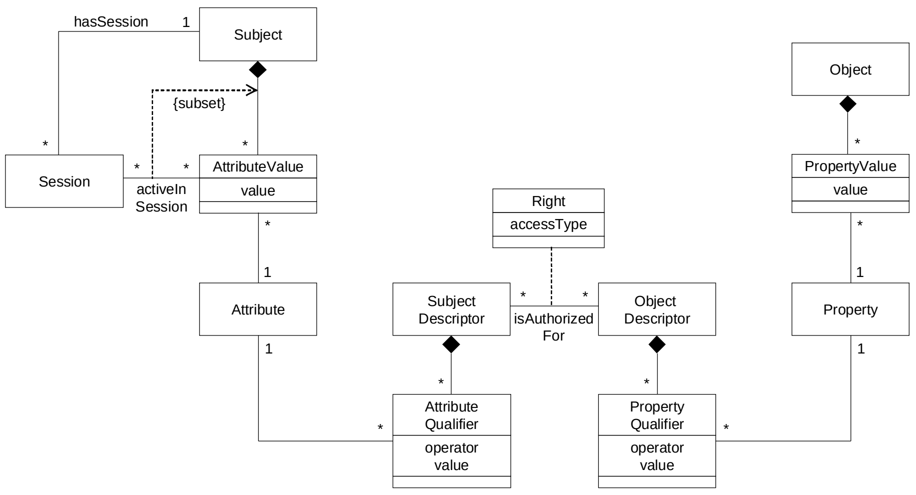

Figure 6. MBAC Pattern with Sessions

- **Consequences:** The administration effort is reduced and flexibility is enhanced by using subject attributes and object properties for authorization. By using sessions the implementation of the least privilege principle is supported, i.e. users have just as many permissions and privileges as they need to perform their current tasks. In addition to these requirements, the session concept might help to enhance user privacy. A user will potentially only want to disclose personal data (e.g. age or sex) if it is indispensable for the task he intends to perform[3]. For example, an e-commerce system selling adult entertainment needs to know the age of a customer before granting him access to its products. If attributes are assumed to be centrally stored and managed, users have only limited control over the use of their personal data. The idea of storing user attributes in attribute certificates rather than in a central database may help to overcome this issue.

- **Known uses:** Again, the .NET framework can use the concept of session in conjunction with their attribute-based model [26].

6. CONCLUSIONS

We have presented a pattern system that provides a structured and unified description of several access control models. After identifying a general authorization pattern and a session pattern, we presented patterns for the well known Discretionary and Role-based Access Control models. In order to address the higher flexibility requirements of open, heterogeneous systems, we finally developed a pattern for a Metadata-based Access Control model (MBAC). These models can be used as a repertory of models to build secure systems and integrated into a secure development methodology [19].

The presented MBAC approach assumes the existence and administration of suitable metadata (i.e. user attributes and object properties). Obviously, access control requirements can only be implemented on the basis of existing metadata, and only if the required attributes/properties are available in appropriate quality. If they have to be collected solely for the purpose of access control, the advantage of reduced administration of course doesn’t hold anymore. We therefore consider desirable to make use of attribute certificates that can be provided and managed by trusted third parties to reduce the metadata administration on the subject side. For the object side metadata from document or content management systems needs to be integrated. Here Semantic Web technologies (e.g. RDF [33]) seem very promising.

Our current work in progress deals with the prototypical implementation of the MBAC model and its integration into the CSAP security module [10], which has been developed within the EU-funded project Webocracy[4]. This

[3] See also the controlled disclosure of personal properties on the Internet in P3P (Platform for Privacy Preferences) [34].

[4] Web Technologies Supporting Direct Participation in Democratic Processes, research and development project in the 5th framework of the European Union (IST-1999-20364).

security module is furthermore being integrated into an enterprise portal system [28], which manages resource metadata by means of RDF. Portlets with different security requirements will be able to use different access control models (RBAC, MBAC) without requiring multiple authentication of the users. The building block approach of the pattern system presented in this paper provides an important basis for the modular design of such a universal security module.

REFERENCES

1. Adam, N.R., Atluri, V., Bertino, E., Ferrari, E.: A Content-based Authorization Model for Digital Libraries. In: *IEEE Transactions on Knowledge and Data Engineering*, Volume 14, Number 2, March/April 2002.
2. Biskup, J.: Credential-basierte Zugriffskontrolle: Wurzeln und ein Ausblick. In: *32. Jahrestagung der Gesellschaft für Informatik e.v. (GI)*, Dortmund, September/October 2002, S. 423-428.
3. Brown, F., DiVietri, J., de Villegas, G.D., Fernandez, E.B.: The Authenticator Pattern. In: *Proc. 6th Conference on Pattern Languages of Programs (PLoP 1999)*, Urbana, IL, USA, 1999.
4. Brown, W.J., McCormick III, H.W., Thomas, S.W.: *AntiPatterns and Patterns in Software Configuration Management*. Wiley, New York, 1999.
5. Braga, A.M., Rubira C.M.F., Dahab, R.: Tropyc: A Pattern Language for Cryptographic Software. In: *Proc. 5th Conference on Pattern Languages of Programs (PLoP 1998)*, Monticello, IL, USA, 1998.
6. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: *Pattern Oriented Software Architecture: a System of Patterns*. Wiley, Chichester 1996.
7. Castano, S., Fugini, M., Martella, G., Samarati P.: *Database Security*. ACM Press, 1994.
8. Clark, D. and Wilson, D.: A Comparison of Commercial and Military Computer Security Policies. In: *Proc. IEEE Symposium on Security and Privacy*, Oakland, April 1987.
9. Dittrich, K.R., Hartig, M., Pfefferle, H.: Discretionary Access Control in Structurally Object-oriented Satabase Systems. In C.E Landwehr (Ed.): *Database Security II: Status and Prospects*, Elsevier Science Publishers B.V. (North-Holland), 1989.
10. Dridi, F., Fischer, M., Pernul, G.: CSAP – An Adaptable Security Module for the e Government System Webocrat. In: *Proc. of the 18th IFIP International Information Security Conference (SEC 2003)*, Athens, Greece, Mai 2003.
11. Dridi, F., Muschall, B., Pernul, G.: Administration of an RBAC System. In: *Proc. Hawaii International Conference on System Sciences (HICSS-37)*, Waikoloa Village, Big Island, Hawaii, USA, Januar 2004.
12. Essmayr, W., Pernul, G., Tjoa, A.M.: Access Controls by Object-oriented Concepts. In: *Proc. of 11th IFIP WG 11.3 Working Conf. on Database Security*, August 1997.
13. Ferraiolo, D.F., Kuhn, D.R., Chadramouli, R.: *Role-based Access Control*. Artech House, Boston et al., 2003.
14. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D., and Chandramouli, R.: Proposed NIST Standard for Role-based Access Control. In: *ACM Transactions on Information and Systems Security*, Volume 4, Number 3, August 2001.
15. Ferrari, E., Adam, N.R., Atluri, V., Bertino, E., Capuozzo, U.: An Authorization System for Digital Libraries. In: *VLDB Journal*, Volume 11, Number 1, 2002.

16. Fernandez, E.B., Pan, R.: A pattern language for security models. In: *Proc. 8th Conference on Pattern Languages of Programs (PLoP 2001)*, Monticello, IL, USA, September 2001.
17. Fernandez, E.B., Larrondo-Petrie, M.M., Gudes, E.: A method-based authorization model for object-oriented databases. In: *Proc. of the OOPSLA 1993 Workshop on Security in Object-oriented Systems*, Washington, DC, USA, October 1993, pp. 70-79.
18. Fernandez, E.B.: Patterns for Operating Systems Access Control. In: *Proc. 9th Conference on Pattern Languages of Programs (PLoP 2002)*, Monticello, IL, USA, 2002..
19. Fernandez, E.B.: Layers and non-functional patterns. In: *Proc. of ChiliPLoP 2003*, Phoenix, AZ, USA, March 2003.
20. Fernandez, E.B., Thomsen, M., Fernandez, M.H.: Comparing the security architectures of Sun ONE and Microsoft .NET, Chapter 9 in Bellettini, C., Fugini, M.G. (Eds.): *Information Security Policies and Actions in Modern Integrated Systems*, Idea Group Publishing, 2004, pp. 317-330.
21. Fowler, M.: Analysis Patterns: *Reusable Object Models*. Addison-Wesley-Longman, Reading, MA, USA, 1997.
22. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley-Longman, New York 1995.
23. Georg, G., France, R., Ray, I.: An Aspect-Based Approach to Modeling Security Concerns. In: *Proceedings of the Workshop on Critical Systems Development with UML*, Dresden, Germany, September 2002.
24. Hays, V., Loutrel, M., Fernandez, E.B.: The Object Filter and Access Control Framework. In: *Proc. 7th Conference on Pattern Languages of Programs (PLoP 2000)*, Monticello, IL, USA.
25. N.N.: *X.509: The Directory – Public Key and Attribute Certificate Frameworks*. ITU-T Recommendation, 2000.
26. LaMacchia, B.A., Lange, S., Lyons, M., Martin, R., Price, K.T.: *.NET framework security*. Addison-Wesley, 2002.
27. Pernul, G.: Database Security. In: Yovits, M. C. (Eds.): *Advances in Computers*, Vol. 38. Academic Press, San Diego et al., 1994, pp. 1-74.
28. Priebe, T., Pernul, G.: Towards Integrative Enterprise Knowledge Portals. In: *Proc. of the Twelfth International Conference on Information and Knowledge Management (CIKM 2003)*, New Orleans, LA, USA, November 2003.
29. Ray, I., Li, N., Kim, D., France, R.: Using Parameterized UML to Specify and Compose Access Control Models, In: *Proceedings of the 6th IFIP WG 11.5 Working Conference on Integrity and Internal Control in Information Systems*, Lausanne, Switzerland, November 2003.
30. Schumacher, M.: *Security Engineering with Patterns: Origins, Theoretical Model and New Applications*. Springer, Berlin 2003.
31. Schumacher, M., Fernandez, E.B., Hybertson, D., Buschmann, F. (Eds.): *Security Patterns*. Wiley, 2004 (to appear).
32. Yoder, J., Barcalow, J.: Architectural Patterns for Enabling Application Security. In: *Proc. 4th Conference on Pattern Languages of Programs (PLoP 1997)*, Monticello, IL, USA, 1997.
33. N.N.: *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation, 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/
34. N.N.: *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*. W3C Recommendation, 2002. http://www.w3.org/TR/2002/REC-P3P-20020416/