6th International Conference on Applied Informatics
Eger, Hungary, January 27–31, 2004.

Access Control Models for Collaborative Applications*

Gyöngyi Bujdosó

In honour of Professor Mátyás Arató
on the occasion of his 70th birthday

Department of Computer Graphics and Library and Information Science
Faculty of Informatics
University of Debrecen

e-mail: bujdoso@inf.unideb.hu

Abstract

There are many models of access control methods used to manipulate the access to shared data in collaborative applications. The advantages and disadvantages of these models vary because while developing these models researchers focus only on certain properties of the functions of workgroups – trying to improve these properties of access control. In this paper, we examine some important access control methods used in groupware systems, and give a possible solution on how to ameliorate certain properties of Capability-Based Security and Role Based Access Control in collaborative works.

Key Words and Phrases: Access control methods, Computer-supported cooperative work, Collaborative applications, Groupware systems

1. Introduction

The problems of management and organization of access for shared resources has a long history. Since the beginning of developing multi-user systems it has been necessary to share certain resources among several users and tasks. Various access control methods have been proposed and used in the field of multi-user operating systems, distributed systems and database systems. Traditional access control models of the operating and distributed systems and database world do not satisfy the needs of collaborative applications. This was stated, e.g., by Greif and Sarin [6] fifteen years ago. From those days work on access control has been done in the Computer Supported Cooperative Work (CSCW) community, but with limited

*Research was partially supported by Tempus (Erasmus) and OTKA No. T032361.

success. A handful of access control models specifically designed for collaborative environments has been published, however, the accomplishment of adequate security for the shared information used and produced is critical for supporting cooperative tasks, where the participants in a task are not equally trusted.

In this paper, we deal with the most important access control methods which are used for groupware applications. Our aim is to examine and compare these methods, and try to find out what their advantages and disadvantages are.

2. Classical Access Control Methods

The history of organization of access for data began with the birth of multi-user systems. In that context the main purpose is to develop highly secured system, where the users are able to hide data from other users. During the evolution of these systems several access control methods have appeared. In this section we deal with those methods which are the most important ones from the point of view of cooperative systems. These methods – or some parts of them – are applied for managing the access for shared data in cooperative systems, and can be classified according to their properties. The traditional access control methods become the base and are used with several other methods which are better for collaborative work.

2.1. Basic problems

In the domain of operating systems, multi-user and distributed systems, database world and groupwares a significant problem consists of allowing several users to reach certain resources which they must or want to share. That requires to define:

- the constraints of simultaneous handling of the shared data,
- the mechanisms for the management and organization of access to these resources.

In the literature one can find several *constraints* which must be taken into account during the development of groupwares. We mention some examples here: confidentiality, responsibility, security, inheritance, integrity, accessibility, flexibility, dynamism, ease-of-use.

By virtue of the essential differences in system behaviors, the constraints that must be satisfied are varied. Some of these constraints are part of each system mentioned above, such as security, confidentiality, responsibility or integrity. Since these requirements are essential and must be satisfied on a high level in long time used operating, multi-user and distributed systems. There are several access control methods which can handle these problems. There are constraints, e.g., flexibility, dynamism and ease-of-use (see Section 3.1) which are not inherited, however, must be satisfied in an efficient collaborative system.

The system of all of these constraints is very sophisticated because there are several relations among them and many have controversial influences on the system, for example flexibility and security or ease-of-use one one hand, accessibility and confidentiality or security on the other. It may be impossible to ensure all of them in a single system on a high level.

2.2. Traditional Access Control Methods

In this section, we examine those access control methods which were the basis for the others and have also been applied recently.

2.2.1. Access Matrix

Classical access matrix is a model of protection (proposed by B. W. Lampson) which was developed for controlling access in operating systems at the end of the 1960's (see, e.g., [10], [14]).

In an access matrix $E = (e_{ij})$, the rows represent subjects (entities wishing to access data), and the columns represent objects (units of data to be accessed). Each e_{ij} entry (cell) of the matrix represents the set of operation that a process – executed by the i-th subject – can perform on the j-th object.

Access matrices are implemented very rarely as matrices because an access matrix can become enormous in size, and most of its cells are empty. Problems concerning groupwares are: classical access matrix is not sufficient for meeting several important collaboration requirements, such as sufficiency of multiple user roles, easy specification, or automation, though it was used by Shen and Dewan ([12]) for controlling acces in a collaborative work (called Suite), using it with several additional types of rights and tools.

We rather deal with its implementations: access control list, and a method that has grown out from the access matrix – the capability-based security.

2.2.2. Access Control List

In the systems using Access Control Lists (ACLs), each object (or resource) – that the users want to access – is associated with a list (ACL) that describes the rights of the users or user groups to this object. The permissions that can be specified by ACLs obviously will vary according to the type of the object the users are interested in. If these objects are files or directories the permissions could be of the type reading, writing, addition, creation, execution, or deletion. These permissions can be negative or positive, and they can also relate to the management of these rights of access and can be one of the following rights: rights of modification of the existing permissions, or rights of giving and revoke permissions for the other users.

In this method each object is associated with a list which makes it possible to specify the relations between the object and the users or user groups, and for each user or user group having particular rights it is necessary to describe all its relations (add, delete, modify etc.) to each object they can use.

Access Control List is a popular implementation of the access matrix in practical systems (see, e.g., [10]). This approach corresponds to storing the access matrix by columns. ACLs are used for organizing access to shared objects e.g. in Java and Microsoft Windows NTFS ([1]), and in collaborative systems such as Basic Support for Collaborative Work (BSCW) (see, e.g., [13]).

ACLs have several *disadvantages*. The implementation of an access control method using ACLs in a system needs a lot of testing. The inheritance of ACLs and of roles cannot be solved in a proper way. The specification of roles is static, this method lacks in dynamism. It needs an administrator who generates the ACLs. In general, the modification of the rights of a role has been done by assigning a new ACL to a system that can result in an ACL system becoming a 'spaghetti'.

2.2.3. Discretionary Access Control

Discretionary Access Control (DAC) methods are based on the concept of *ownership*: individual users are owners of objects, so they have complete discretion over who should be authorized to access the object and in which mode. This control access is based entirely on the identities of users and objects.

DAC was developed for solving security in systems of commercial and government organizations. DAC is a tool of restricting access to objects based on the identity of subjects and/or groups to which they belong. In this mechanism, for each object, a particular user or a group of users has the authority to distribute and revoke access to that object. The controls are discretionary in the sense that a user or process given discretionary access to information is capable of passing that information along to another subject.

An important *deficiency* of DAC mechanism is that users are responsible for securing their own data, i.e., users are authorized to decide (or forget) not to secure information, making it available to inappropriate access.

Let us speak about the following scenario. The first user who is the owner of an object gives all of his rights to this object to one of his friends (second user). After that the second user has the same rights to this object as the original owner, so he can transmit all of his rights to the object to a third user. One month later the first user revokes all of the rights of the second user to the object, but he does not know anything about the rights of the third user, consequently he cannot revoke them from him.

Another *disadvantage* of this method is that the inheritance of rights for data is not solved. Users with a 'read' right to a file can copy and paste any part of the file without the original access rights causing the collaborative work to be dangerous.

2.2.4. Mandatory Access Control

Mandatory Access Control (MAC) method is implemented by adding 'sensitivity labels' to each file, window, program, host, network, and devices. For anyone accessing the system, security administrators indicate the level of trust or job responsibility by assigning a 'clearance' that sets one sensitivity label, or the upper and lower bounds of a set of sensitivity labels at which the user can work.

This mechanism has been developed for achieving multi-level security in military systems (see, e.g., [8]), so the most important issue was to achieve a high level security in the system. For example, the Trusted Solaris software uses MAC (see, e.g., [16]) for granting secure access to sensitive information, programs, or devices in the system.

Controlling access with this method gives us a secure system, but it does not support well group works. It has several reasons. Some examples: labelling of information occurs automatically, ordinary users are not allowed to change sensitivity labels or clearances; changing clearances during a session is almost impossible; the one-directional flow of information in a lattice there are no covert channels through which information can flow in prohibited ways ([9]).

3. Access Control Models for Collaborative Systems

Collaborative environments introduce new requirements for access control, which cannot be met by using existing models developed for non-collaborative domains. Presently there is no general model for controlling access that could be capable of satisfying all the needs of collaborative works.

In this section we are dealing with the special requirements of cooperative works and those access control models that have capability to satisfy some of the new requirements of collaborative systems. These models could be the basis of several new models that will be able to satisfy more and more groupware requirements.

3.1. Special Needs of Collaborative Systems

Collaborative systems have many requirements that exist in multi-user, shared and distributed systems. In addition, groupware systems introduce new requirements in the domain of access control that are not satisfied entirely by the traditional access control models developed for traditional systems (see, e.g., [2], [3], [11], [12]).

The following issues are generally mentioned in relation to supporting access rights in collaborative systems:

- Collaborative systems need access rules to shared data which are able to evolve in a *dynamic* manner.
- *Access rights* should be given to *roles* (e.g., mathematician, project leader, supervisor), rather than to individual persons.
- *Users* take on assigned *roles.*
- In such a system, users should take *multiple roles.*

- Users should have capabilities for *playing their different roles* in a scenario, they should change these roles *dynamically* during different phases of collaboration ([12]).
- Roles must be *flexible*, sometimes role allocation has to be determined at runtime, as requests for access are made ([4]).
- When users take multiple roles in a scenario, one of the following concepts can be chosen: The user's roles should operate as a *sum* of the permissions of these roles (e.g. [7], [12]); or the user's rights can be the *intersection* of the permission defined by the user's roles.
- Cooperative systems need abilities to define *positive* and *negative* access rights. The existence of negative rights is necessary in a system that may contain hierarchical group structures – because situations when some users (or groups of users) must be excluded from some rights may appear.
- Certain access rules must be sensitive to the different levels of collaboration and to the relations of the users. These rules are called *collaboration rules.*
- The operations (beside the traditional ones such as reading and writing) whose results of which can affect multiple users should be protected by *collaboration rights* ([12]).
- Users in collaborative systems need capabilities for *delegating* access rights from one user to another, with the possibility of *revoking* the delegated rights ([7]).
- These systems should be '*easy-to-use*', i.e. users should be able to specify access definitions easily ([12]), and roles must be created in an easy way.
- The *inheritance* of access rights has a very important role in collaborative systems concerning the roles of users and also the protection of data.

It is clear that to develop a collaborative system which satisfies all of the necessary requirements is a very difficult task. The problem still exists, and our aim is to find the best compromise possible for the solution.

The next two subsections offer a brief survey of those access control models that can be made suitable for controlling access in groupware systems.

3.2. Capability-Based Security

Capability-Based Security (CBS) is a more modern exposition of capabilities which "emphasises the elegance and expressive power of capability-based security" as stated by Smith, Hixon and Horan in [15]. This method was used for controlling access rights in collaborative systems the first time in 1998, in a system called *Kansas* (see, e.g., [15]).

CBS is based on *capabilities.* In this model, there are three types of objects: the *protected object*, the *user object* and the *capability object.* A capability object allows us to define the kind of operation can be performed by a given user on a given object or type of object. Each protected object maintains a table of capabilities. This table is like a list of key–value pairs, where the key is a descriptive string name of the operation allowed to perform on the object, and each value is a capability. Each user who wants to access a protected object is a 'user' object, and each user object maintains a capability set that determines the operations the user can perform, and retains a reference to the protected object. In practice, when a user wants to access a protected object, the system compares the 'values' of the capability table for the protected object and the 'values' of the user's capability set. The user can perform an operation on the protected object if the same values exist in the capability set and in the table of capacities.

As we mentioned above, capability objects were first used to control access in operating systems. It was used as an implementation of the Access Matrix in practical systems (see, e.g., [10]): In an access matrix, the list of objects together with the operations allowed on those objects is the capability list for a subject ([14]). This approach corresponds to storing the access matrix by rows.

This method has several *advantages* concerning the new requirements of groupware systems: this model allows the delegation of authority by transferring capacities from one user to another. This model allows constructing a flexible and dynamic system. It has a mechanism of inheritance that allows managing object groups in an identical way. This mechanism is based on the concept of object, thus its application and implementation for control access mechanisms in distributed object-oriented systems (such as Java RMI or CORBA) is remarkable and quite easy. Its inheritance mechanism allows us to get some of the benefits of negative rights.

On the other hand, this mechanism has serious *disadvantages.* One of its most serious disadvantages is the lack of security, i.e., the possibility of transferring capabilities from one user to another and from one object to another, in general, diminishes system security.

3.3. Role-Based Access Control

With Role-Based Access Control (RBAC), access decisions are based on the *roles* that individual users play as part of an organization. Users take on assigned roles, a role can represent competencies for performing various tasks (such as that of a mathematician), and it can be an authority or a responsibility (such as manager of a project). Permissions are linked with roles, and users are connwcted to roles (based on their responsibilities and qualifications). In RBAC, there is a hierarchy of roles, which can help incorporate further requirements of collaborative systems.

The concept of role came originally from organization theory. RBAC was first applied in multi-user and multi-application on-line systems in the 1970s. It was used, e.g. in Novell NetWare, Windows NT, SQL3 and CORBA. It can also be found in groupware systems such as Intermezzo'96 (see, e.g., [4], [5]).

This mechanism has many *advantages*. Its right to assign mechanism (based on the membership in roles) greatly eases the administration of authorization. Users can be easily reassigned from one role to another. The hierarchy of roles helps realize a quite good inheritance of rights. RBAC supports static, dynamic and aggregate roles. It has great flexibility, although it is capable to perform security on a high level. Dynamic roles allow evolving the user's rights dynamically.

However, we can also find some *inconveniences*, for example in the domain of 'ease-of-use', and on the level of collaboration. Creating a new role keeping in view all of the requirements of the system is not easy and has two inconveniences: the implementation of roles for a real system is expensive; moreover it is a more difficult task during collaboration. RBAC makes it possible to use dynamic roles, but this dynamism is not sufficient for performing flexible cooperations.

4. Comparison of the Models

In this section we examine and compare these methods relating to some important requirements of collaborative applications.

When we compare the methods discussed above we examine them from the point of view of the traditional requirements such as confidentiality, responsibility, accessibility, security, and of some new requirements, such as flexibility, dynamism, inheritance, sensibility on the levels of collaboration, giving and revoking roles, as well as creating roles.

4.1. Traditional Access Control Methods

When relating to the traditional requirements we can see that DAC and MAC have quite good properties concerning security, confidentiality and responsibility, but they do not allow to develop a flexible system, and also they have problems with accessibility. Acccess Matrix applied in Suite does not support a dynamic evolution of roles, it has problems concerning the management of inheritance and flexibility, and as well as the support of the levels of collaboration. ACL (applied, e.g., in BSCW) has several problems with almost all of these requirements. It does not support well security in a dynamic, evolving collaborative work.

Hence, traditional access control methods do not support well the new requirements of groupware systems. In these models, dynamism is on a low level, it is hard to give and revoke roles in a proper way, they have no good collaboration level sensitivity, and neither inheritance nor flexibility is solved well.

4.2. Access Control Methods for Groupwares

In view of the traditional requirements we can say that CBS, applied in Kansas, is not good enough. Security is on a very low level in it, responsibility and confidentiality have to be ameliorated. RBAC used in Intermezzo has better properties concerning accessibility and security.

Examining the new requirements, we can say that these models have better properties than the traditional methods. CBS has better properties in the domain of giving and revoking roles appropriately. CBS makes it possible to develop dynamic and flexible systems where inheritance can be solved on a higher level. By RBAC we can establish a secure and dynamic access control mechanism. RBAC can support well inheritance.

If we ameliorate the security of CBS, it may turn out to be a better model for controlling access in groupwares. If we guarantee a better security of right delegation in RBAC and solve the problems of flexibility, we get a new access control method which fits collaborative systems better.

5. Electronic Certificate could be a Solution

Finding the weaknesses and shortcomings of existing access control methods has led us to suppose that the technology of electronic certification could eliminate some deficiencies of Capability-Based Security and Role Based Access Control.

Electronic certification has several useful characteristics which might prove beneficial to certain properties of our methods. This new technology provides us an opportunity to improve security, responsibility, confidentiality, flexibility and inheritance in access control models.

Concerning CBS, the capability of electronic certificates for protecting information could ameliorate the security, the confidentiality and the responsibility of the system combining it with the 'capabilities' of the CBS method. Concerning RBAC, the part of a certificate that contains access information (e.g. restriction) may attach negative rights to roles in RBAC. This combination might improve the sensibility on collaboration levels, and the flexibility and dynamism of collabotarion.

6. Conclusions and Perspectives

In this paper, we have examined some important access control models used in collaborative systems, and some other methods which partially formed the basis of the newer methods. We can say that the later methods such as CBS or RBAC are better for controlling access in collaborative applications but they do not satisfy all of the requirements of groupwares.

Our objectives has been to find a technique that can eliminate the shortcomings of the existing methods. It seems that electronic certification may help us in this process, because, in our opinion, electronic certificates could be the tools for ameliorating security in CBS on one hand, and dynamism, fexibility and sensitivity in RBAC on the other. Our next goals are to elaborate a new access control method for collaborative applications that combines the CBS and RBAC methods with the technology of electronic certification for generating a more adaptable technique, and developing their prototypes.

7. Acknowledgements

Part of the research described in this paper was done by the author while on DEA studies at Laboratoire des Systèmes Communicants, Université de Savoie, Chambéry, France. We wish to thank Stéphane Talbot for his support and helpful comments on this work.

We wish to thank Professors Mátyás Arató, Katalin Bognár and László Kozma for their valuable remarks and for their support, which enabled the author to study at the Systèmes Communicants Laboratory.

References

[1] *Access Control Methods,* Center for Information Security Technology of Science Applications International Corporation, `http://research-cistw.saic.com/cace/`.

[2] Atallah, S. B., Balter, R., Kanawati, R. and Riveill, M., Collecticiels syncrones : analyse des besoins et étude des architecture, *Techn. Rep.,*

[3] Dewan, Prasun and Shen, HongHai, Flexible meta access control for collaborative applications, *Proceedings of the ACM CSCW'98 Conference, 1998,* ACM, 1998, 247–256,
`http://www.cs.unc.edu/~dewan/`.

[4] Edwards, W. Keith, Policies and Roles in Collaborative Applications, *Proceedings of the ACM Conference on CSCW (CSCW'96),* ACM, Cambridge, MA, USA, 1996, 11–20.

[5] Edwards, W. Keith, Representing Activity in Collaborative Systems, *Proceedings of the HCI INTERACT'97 Conference,* Chapman & Hall, 1997.

[6] Greif, I. and Sarin, S., Data Sharing in Group Work, *Proceedings of the ACM Conference on Computer Supported Cooperative Work,* ACM, Austin, Texas, 1986.

[7] Kanawati, R. and Riveill, M., Access Control Model for Groupware Applications, *Proceedings of the HCI'95: People and Computers Conference,* School of Computing and Mathematics, University of Huddersfield, UK, 1995, 66–71.

[8] McLean, J., A Comment on the 'Basic Security Theorem' of Bell and LaPodula, *Information Processing Letters,* **20** 2 (1985), `http://chacs.nrl.navy.mil/personnel/mclean.html`.

[9] Sandhu, Ravi, Access Control: The Neglected Frontier, *Proceedings of the First Australasian Conference on Information Security and Privacy,* Wollongong, Australia, 1996,
`http://www.list.gmu.edu/confrnc/acisp/ps_ver/acisp96.ps`.

[10] Sandhu, Ravi and Samarati, P., Access Control: Principles and Practice, *IEEE Communications,* **32** 9 (1994), `http://www.list.gmu.edu/journals/commun/pdf_ver/i94ac.pdf`.

[11] Schmidt, Kjeld and Simone, Carla, Mind the gap! – Towards a unified view of CSCW, *Proceedings of COOP 2000, Sophia Antipolis, France,* INRIA, Sophia Antipolis, France, 2000, `http://www.cti.dtu.dk/CSCW/CSCWpubl.html`.

[12] Shen, HongHai and Dewan, Prasun, Access control for collaborative environments, *Proceedings of the ACM CSCW'92 Conference, 1992,* ACM, 1992, 51–58, `http://www.cs.unc.edu/~dewan/`.

[13] Sikkel, Klaas, A Group-Based Authorization Model for Cooperative Systems, *Proceedings of the ECSCW'97 Conference, Lancaster, 1997,* Kluwer Academic Publishers, Dordrecht, 1997, 345–360.

[14] Silberschatz, Abraham, *Operating System Concepts,* 4th edn, Addison-Wesley, Reading, 1991, 431–457.

[15] Smith, Randall B., Hixon, Ranald and Horan, Bernard, Supporting flexible roles in a shared space, *Proceedings of the ACM CSCW'98 Conference,* ACM, 1998, 197–206,
`http://www.acm.org/sigchi/cscw98/program/onepage.html`.

[16] *Trusted Solaris Features: Mandatory and Discretionary Access Control,* 2000, `http://www.sun.com/software/solaris/trustedsolaris/7/ts_feature_macdac.html`.

Postal address

Gyöngyi Bujdosó
Department of Computer Graphics and
Library and Information Science
Faculty of Informatics
University of Debrecen
Debrecen, 4010, P.O. Box 12, Hungary