Data Protection and Data Security Concept for Medical Applications in a Grid Computing Environment

Dissertation
zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultäten
der Georg-August-Universität zu Göttingen

vorgelegt von
Yassene Mohammed
aus Kuwait City, Staat Kuwait

Göttingen 2008

D7
Referent: Prof. Dr. Otto Rienhoff
Korreferent: Prof. Dr. Jens Grabowski

Tag der mündlichen Prüfung:

Summary

This thesis is addressing the authorization problem in healthgrid environments, which is still an obstacle in front of using grid computing for medical applications. Current grid authorization systems depend on the traditional access control approaches, in which users are granted static rights. These approaches were designed for closed systems, i.e. the computing system itself enforces the security policy. Grid computing is characterized as open computing system, which means there exist different authorities and multiple agents participating in the authorization process. Therefore, the adoption of classical access control models in a grid computing environment is not adequate.

In this work, access control in grid computing is tackled from a new perspective. The authorization process is handled as a multi-player game and a new authorization model – the Grid Usage Control (G-UCON) – is developed as a solution outline for a suitable access control model for grid computing environments.

G-UCON utilizes the notions of game theory, multi-agent systems, and open systems in the design of a new access control model. These are the minimal requirements for a basis access/usage control model for a grid computing environment. The Alternating-time Temporal Logic is used to write the G-UCON specifications because it is more suitable to capture the specifications of games over open systems.

Various examples at the end of this work demonstrate how G-UCON can capture different complex situations which appear in healthgrid environments. Using G-UCON, special authorization requirements in the health applications can be modeled. Examples are: disclosure risk control and maintaining k-anonymity. Currently, there is no tool available which can validate the special grid delegation mechanism. Therefore, a formal validation of G-UCON is left for future work. Nevertheless, it is a common practice to supply new authorization models with examples until a formal validation and/or an implementation appears which normally happens some years later.

Zusammenfassung

Diese Arbeit befasst sich mit dem Autorisierungsproblem in HealthGrid-Umgebungen, welches immer noch ein Hindernis vor der Nutzung des Grid-Computings für medizinische Anwendungen ist. Die Autorisierungssysteme für Grid-Umgebungen nutzen die traditionellen Ansätze der Zugriffskontrolle, in denen jedem Benutzer statische Rechte gewährt werden. Diese Ansätze wurden ursprünglich für geschlossene Systeme entwickelt, d.h. das Computing-System setzt die Sicherheitsrichtlinien selbst durch. Grid-Computing ist ein offenes System, d.h. verschiedene Domänen bzw. mehrere Agenten nehmen am Autorisierungsprozess teil. Daher ist die Nutzung klassischer Autorisierungsmodelle in eine Grid-Computing-Umgebung nicht angemessen.

In dieser Arbeit wurde die Zugriffskontrolle im Grid-Computing aus einer neuen Perspektive behandelt. Der Autorisierungsprozess wird im Rahmen der Spieltheorie behandelt, und ein neues Zugriffskontrolle-Modell wurde entwickelt. Dieses Modell – das Grid Usage Control Modell (G-UCON) – ist ein Lösungskonzept für ein angemessenes Zugriffskontrollmodell für Grid-Computing-Umgebungen. G-UCON verwendet die Prinzipien der Spieltheorie, der Multi-Agenten-Systeme und der offenen Systeme zum Aufbau des neuen Zugriffskontrollmodells. Diese sind die minimalen Anforderungen für ein Grid-Computing Zugriffs- bzw. Nutzungskontroll-modell. Die „Alternating-Time“ temporale Logik wurde für die Entwicklung der G-UCON Spezifikationen benutzt, weil sie geeignet ist, spieltheoretische Spezifikationen offener Systeme zu erfassen.

Mehrere Beispiele am Ende dieser Arbeit zeigen, wie G-UCON komplexe Zustande erfassen kann, die in HealthGrid-Umgebungen entstehen könnten. Die besonderen Anforderungen an die Nutzungskontrolle in den medizinischen Anwendungen können anhand des G-UCONs modelliert werden. Beispiele hierfür sind die Re-Identifizierungsgefahr bzw. die Gewehrleistung der k-Anonymität. Derzeit steht kein Validierungstool zur Verfügung, das speziellen Grid Delegationsmechanismus modellieren kann. Eine formale Validierung der G-UCON Spezifikationen sollte deswegen eine zukünftige Arbeit sein. Dennoch ist es üblich, dass neue Autorisierungs-modelle mit mehreren Beilspielen bestätigt werden, bis eine formale Validierung bzw. eine Umsetzung erscheint, die normalerweise einige Jahre später passiert.

Acknowledgement

I would like to express my deepest appreciation and thanks to Professor Jens Grabowski for supporting me and for his invaluable advices on how to improve this dissertation. Cordially I thank Professor Dieter Hogrefe for having me as a member in his group and for his support, understanding and backup. Many thanks go to Professor Robert Schaback for his support and advices on how to improve this dissertation. My gratitude goes to Professor Ulrich Sax for his help, kindness, support, all the fruitful discussions, and for having me in his group. I would like to express my deepest gratitude to Mrs. Ursula Piccolo for all the help I received from her over the last few years. Finally, I would like to address my sincere thanks and appreciations to my supervisor Professor Otto Rienhoff for his unlimited support and vital encouragement on the scientific and private levels during the different phases of my research and throughout the writing of this dissertation.

Contents

1. Introduction and Motivation

Data protection in biomedical application needs to be enforced more strictly than in other application areas [1, 2]. Since the 1960s, data protection of personal health information was and is still of high concern. The details of where information flows, who has access to the data and for what purpose are of major importance. Considering this and bearing in mind that in grid computing it is not only simple data sharing, but rather it is the sharing of distributed resources like algorithms, storage, computing power, etc., it is necessary to study closely the aspects of grid security and to find suitable solutions to enforce it.

The privacy levels of biomedical data

Biomedical data are various, contain different information and could be categorized into different levels of privacy. It can be information about (1) Population: epidemiological data, (2) Diseases: clinical practice data, clinical trials, (3) Patient's data: health record, clinical history, physical exams, lab/imaging studies, (4) Organ/tissue: pathology, (5) Cellular: histology, or (6) Molecular: genetic test results and genomic data [3]. Having these data online with the suitable tools to connect, combine and analyze them brings data protection and security to new challenges.

The disclosure risk challenge

Phenotypic data in a patient electronic medical record can lead to identify for example whether the subject has a particular infection or not. Analyses methods for such identification can be simple statistical procedures, like in [4], or machine learning systems – artificial neural network – like in [5]. Anonymizing procedures are not enough for protecting the data without loosing the scientific value of this data. For instance, sharing high-resolution imaging datasets online may be risky; a full reconstruction of the face using computerized 3D techniques is indeed possible [6-8]. Sharing of medical 3D imaging datasets has already been reported in some pre grid environment applications [9, 10], anyhow, the risk of this sharing is not well studied so far. The major problem is that we still cannot enforce a dependable security policy in grids; i.e. we cannot assure that administrators, developers, or other staffs do not have an access to the medical data.

A more complicated problem for data protection in a grid computing environment is the correlating of two non-identifying datasets, which together they become explicit identifying set. For instance, risky is the linking of medical datasets to quasi-identifying data, like ZIP-code, date of birth and gender, which may indeed lead to re-identifying the subject/patient. Although the problem is not new, in a grid environment it takes new dimensions. The distributed nature of the grid resources and entities make it difficult to enforce data protection policies, this increases the disclosure risk [11]. In this context, we have to keep in mind the vision about grids "being the internet of the future" – regarding availability of data and services [12]. The most common example for this is correlating genomic and phenotypic data. Interests in such studies using especially machine learning methods become intense [13, 14]. Privacy and information sensitivity become blurred concepts when talking about genomic data [15].

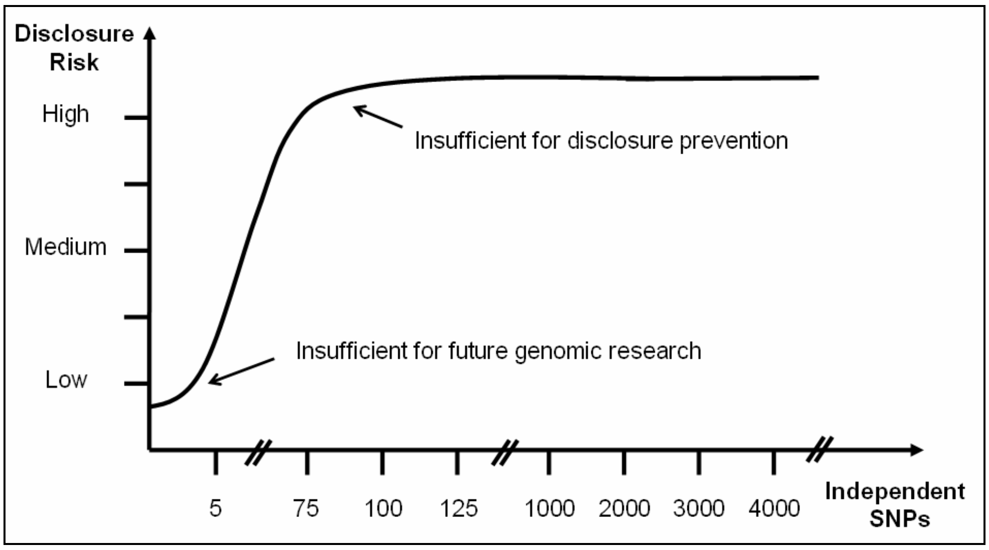


Figure 1: A schematic graphic showing the tradeoff between the number of SNPs revealed in a database and the ability to protect privacy (modified and corrected1 from [16]).

Studying stretches of DNA that have been found to harbor Single Nucleotide Polymorphisms (SNPs, variations occur in the genomic data) and understanding how genetic variations are associated with a disease trait, a heritable disease or with a response to drugs and medical treatment are of increasing interest to

[1] Privacy has a broad meaning and differs between nations, countries, regions and even individuals; it is rather difficult to measure it. It is better to measure disclosure risk, which is an already coined and established measure.

researchers. A grid environment is very convenient for this kind of research [17, 18]. Lin et al. show that one needs only 74 SNPs to re-identify the subject (see Figure 1). With thousands of SNPs in the human genome, it is difficult to imagine disclosure control methods that can protect privacy while maintaining the scientific value of the data [16]. Chromosome 21, which is one of the shortest chromosomes in human contains alone 4563 SNPs [19]. Turakulov and Easteal found in their statistical study on three populations, Afro-American, Asian and Caucasian that as less as 65 random SNPs are required for identifying the subject's population [20]. Hence having genomic and patient records data connected and correlated implies immediate risk for the patient's data protection. The disclosure risk in such studies stays high, even though anonymizing techniques are used.

The data protection challenge in a healthgrid environment

The patient's permission to access her data should be granted in a grid environment, as she is the natural owner of this data. Offering transparent full access for researchers to the data may harm the patient's privacy. The questions, who has access permissions when for which data, which dataset is to be used and where it is stored, as well as when to update this dataset are data protection aspects, which do not only need to be answered, but also to be enforced. In a grid environment, this applies not only for data but also for services that manipulate and process the data. Generally, in Service Oriented Architectures (SOAs) like grids, authorization is about granting rights to use the different resources in grids and not only to read or write data.

In this context, a correct and functional authorization system for grids is the most urgent problem in front of using grid computing for medical applications, especially if we intend to grant access to use sensitive datasets, like patient data. Traditionally, in closed systems where the service provider knows the user, access control focuses on the digital resources; i.e. the access is driven by rights granted to the user to access an object or a service [1, 21-27]. The problem of authorization for these systems is well solved. Distributed systems use extensions to establish trust (authentication) or to communicate the capabilities of the users. In modern dynamic and distributed computing environments like grid computing, datasets and digital information are to be used and stored at various locations, thus have to be protected regardless of the user location and the information location. In contrast to the traditional approach, in the grid one will send jobs (data and algorithms) to be processed on a machine that belongs to somebody else. These jobs carry information about the processed datasets, link datasets to each other, and link datasets to the suitable processing methods. All these increase the disclosure risk.

When using grid computing for medical applications, it is necessary to identify the needed level of security and data protection that fulfill the legal requirements as well as to identify eventually new legal requirements needed to consider new aspects raised by adapting the new technology. Aside from the acceptance by the community, adapting new IT-technology in the medical sector starts normally by identifying and dealing with the shortcomings. Because data protection in the health sector has its special legal framework, these shortcomings should be addressed from two angles of view: the shortcomings of the current data security and data protection legal framework regarding grid technology, and the shortcomings of current grid middlewares regarding the data security and data protection requirements.

As a summary, the data protection challenges in grid computing include (but are not limited to):

- Realizing mechanisms to organize who has access on which resources, for what purpose, and for how long.
- As grid computing is designed to be flexible, it should be possible to add/change/update resources including data dynamically anytime.
- Problems arise about the ownership and responsibility for the resulting data when correlating and processing different datasets. In this context, intellectual property rights and accountability should be considered.
- A known risk in medical data protection generally, and new in medical grid data protection particularly, results from data types with high predictive value, for example genomic data combined with clinical records data [11, 28, 29]. Here, rules for combining datasets, for accessing the tools as well as the resources should be defined and implemented.

The objectives of this work

The purpose of this work is to address data protection and data security for medical application in the grid, namely

- to identify any shortcomings concerning the use of grid computing for medical applications (see section 4.1 on page 67), and
- to identify the most urgent problem to be handled and to develop a solution proposal for this problem (see subsection 4.1.3 on page 82).

Among the different identified shortcomings, the authorization problem appears to be the most urgent one. This work tackles the authorization problem in grid computing environment and handles the following questions:

- What are the additional requirements of a functional authorization system for a grid computing environment? (see section 4.1 on page 67 as well as subsection 4.1.2 on page 73)
- Are these requirements different from the existing solutions? And if yes, why and how? (see subsection 4.2.1 on page 84 and subsection 4.2.2 on page 87)
- What is the origin of the authorization problem in grid environments and how to handle it? (see section 3.3 on page 49, section 4.3 on page 92, and chapter 5 on page 113)

2. A Primer to Grid Computing and Access Control Models

2.1. What is grid computing?

2.1.1. A definition of a grid computing environment

Grid computing is becoming the solution for researchers looking for vast storage and computing capacity, for sharing programs and algorithms. “A computational grid is a set of computing elements and data storage elements, heterogeneous in hardware and software at geographically distant sites which are connected by a network and use mechanisms to share resources as computing power, storage capacity, data” [30] and algorithms. Comparing it with the internet, grid computing goes one step farther in sharing also computing power, storage, applications and algorithms beside sharing information. Similar to semantic web, there is the semantic grid. The semantic web is an extension of the current web in which information is given a well-defined meaning, better enabling computers and people to work in cooperation, i.e. embedding knowledge alongside information. The semantic grid is an extension of the current grid in which information and services are given a well-defined meaning, better enabling computers and people to work in cooperation [12, 31].

Grids are virtual pools of resources rather than computational nodes. Although current systems focus on computational resources (CPU cycles and memory) [32], grids operate on a wider range of resources like storage, network, data, software but also on graphical and audio input/output devices, sensors and so on [33, 34]. All these resources typically exist within nodes that are geographically distributed in multiple administrative domains. Precisely spoken, “the grid is a virtual hypothetical concurrent machine, which is constituted of a set of resources taken from the resources pool” [35].

The high performance computing, which started in the 1970s mainly for physical applications, could be considered as the beginning of sharing computing power. Being developed for the natural and applied sciences, the demand for security and data protection was low. The new grid computing environments (i.e. middlewares) inherit this low level security trait, maybe because these middlewares were developed again from the beginning primarily for the physics community [36].

The goals of using grid infrastructures for medical applications are not only to provide researchers and physicians with access to computing power and to a broad spectrum of data for analysis, but rather to provide the tools to connect different datasets and to perform the needed analyses regardless where the resources are located. Genome Wide Association Studies and Sequence Analysis are in this context typical grid applications [17, 18, 37]. The key developers of the grid technology – the Globus Alliance [38] – are benefiting currently from cooperating with the biomedicine community in order to develop suitable middleware for this application area. Simultaneously, they intend to gain experience, how grid computing could and should be developed regarding important aspects like data protection or like processing distributed sensitive data (patient datasets) [39].

2.1.2. The particularity of grid computing compared to other distributed systems

In traditional resources/services access control, the user has a direct connection to the resources, which will authorize the user or block her. What distinguishes grids is that, unlike conventional distributed systems (like cluster computing), users and resources appear differently at the virtual and at physical levels. This requires an appropriate mapping to be established between them [35]. Therefore, grid technology uses agents for user and resources mapping. The resource mapping agents (brokering systems) are responsible for mapping a user's request to a suitable resource, which is available at the request time point. The user mapping agents will in turn map the requests arrived from the brokers to the local nodes as jobs from local users. Semantically spoken, "the inevitable functionalities that must be present in a Grid system are resource and user abstraction" [35].

Nemeth and Sunderam characterized grid computing and presented formal definition expressed in Abstract State Machine (ASM) to describe grid computing and to distinguish such a computing environment [35]. The existence of resource mapping agents and user mapping agents is the main characteristic, which distinguish grid systems from other systems. These agents play an important role in the authorizing process of the user. Keeping in mind that the user delegate her (or part of her) privileges to a special process to work on her behalf, these agents play a role not only in authorizing the user but also in authorizing processes.

To capture the notion of these two kinds of agents, Nemeth and Sunderam defined two processes: *CanUse* and *CanLogin*. If *CanLogin: USER* × *NODE* → *{true,false}* evaluates to true, it means that user has a credential that is accepted by the security mechanism of the node. It is assumed that initiating a process at a given node is possible if the user can log in to the node. *CanUse***:** *USER* × *RESOURCE* → *{true,false}* is a similar logic function; if it is true, the user is authentic and authorized by an authorizing mechanism to use the abstract given resource defined in the request. While *CanLogin* directly corresponds to the login procedure of an operating system, *CanUse* is a new concept of grid systems and corresponds to an authorizing process to determine what the user is allowed to do and redirect her request to the suitable nodes and resources [35]. In other words: *CanUse* is a brokering function and *CanLogin* is a local account access control function.

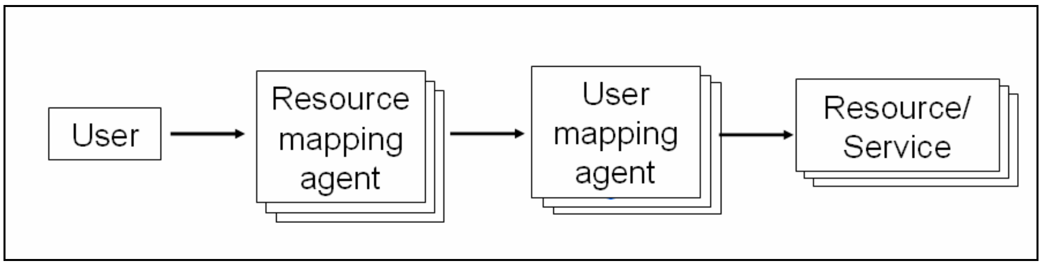


Figure 2: A simplification of the resources access process in grids. The arrow means "communicates".

This results in particular difficulties for authorization, namely:

- The service does not know the user; it only knows the local account to which the user is being mapped to. This leads for example to an auditing problem since the log files and activity protocols refer to the local accounts. It also results in fine granular access rights problems because the service grants access according to local accounts, not the real users.

- The grid user cannot choose explicitly the provider of a service; typically, the user's request includes only an abstract description of the needed service, not the address (location, administrative domain) of the service. Sometimes, this is a problem for user's intellectual property rights because she cannot prevent the monitoring of her work through the service provider. Similarly, the user cannot decide what information about herself will be communicated to the service provider. This may result in a problem about maintaining the “Informational self-determination” right.

- The user, the service and the agents may have conflicts of interest; all entities in the grid have their own policies and interests. The grid community is talking about the problem of (un)suitable mapping: “The Grid system should follow each domain’s security policies and also may have to identify users’ security policies.” [40]. The new “buzz” in the grid community about developing the grid Service Level Agreements [41] is an attempt to solve this problem.

- The usage of a resource/service is temporary and not static; i.e. the users’ rights to use the resource/service are in turn temporary. This is also applicable for services from a specific provider in the grid. In this regard, there is a practical problem in isolating usages of different users regarding time (for example to avoid a memory espionage [42-44]). Problems appear also regarding saving profiles for later usages. In other words: time is not considered in the authorization decision.

The grid community efforts to develop grid access control solutions were for most attempts to accommodate the technology to well known and accepted access control models. Mostly this approach malfunctions. To understand this better, next section explains and discusses what is a security model and what models were adapted for grid computing.

2.2. What is an access control model?

The refinement process of a higher level of abstraction to a more detailed description is an established engineering method, especially when dealing with complex problems. The move from the question “What is needed” to “How to implement these needs” is essential in developing a solution for a complex problem. This applies also in finding the characteristics of complex systems like the grid [35] or like the authorizing process in the grid.

It is well accepted that security engineering follows a similar refinement process. A layered approach to security engineering “will typically take the form of: threat model-security policy-security mechanisms” [1]. Sandhu proposed in 2000 a more detailed layered approach to security engineering; i.e. the Objectives, Models, Architectures, and Mechanisms framework [45]. This approach gained a good acceptance [46-49] for its simplicity and ability to describe already implemented access control models like Role Based Access Control [45].

The Objectives, Models, Architectures, and Mechanisms (OM-AM) framework consists of four layers (see Figure 3). **The objective layer** captures the high-level security requirements of a system (policies or goals). It is driven by anticipated threats and goals, and is usually written in natural language.

The model layer “decomposes policies into abstract terms that can be analyzed and mapped into implementable entities”. “This often takes the form of formal, rigorous mathematical descriptions, but sometimes precise use of natural language is sufficient” [46]. Access control models are usually written to precisely capture the security properties that access control should adhere to; i.e. bridging the rather wide gap in abstraction between policies and “mechanisms” [50][2]. “Users see access control models as an unambiguous and precise expression of requirements; vendors and system developers see access control models as design and implementation requirements” [50]. The Next section (2.2.1) describes some examples of access control models like Role Based Access Control or Discretionary Access Control.

The architecture layer describes a high-level security design in terms of the major components of the system(s) (e.g. servers, brokers, databases, middleware, etc.) and their interactions and relationships.

The mechanism layer focuses on the means and techniques to implement the security design. For distributed systems, these may include network protocols, credentials, or tickets. The mapping between these layers is many-to-many; a

[2] This report was written as an abstract point of view to which no “architecture layer” is needed to be considered.

model can be supported by multiple architectures and an architecture can support multiple security models [25, 45, 46].

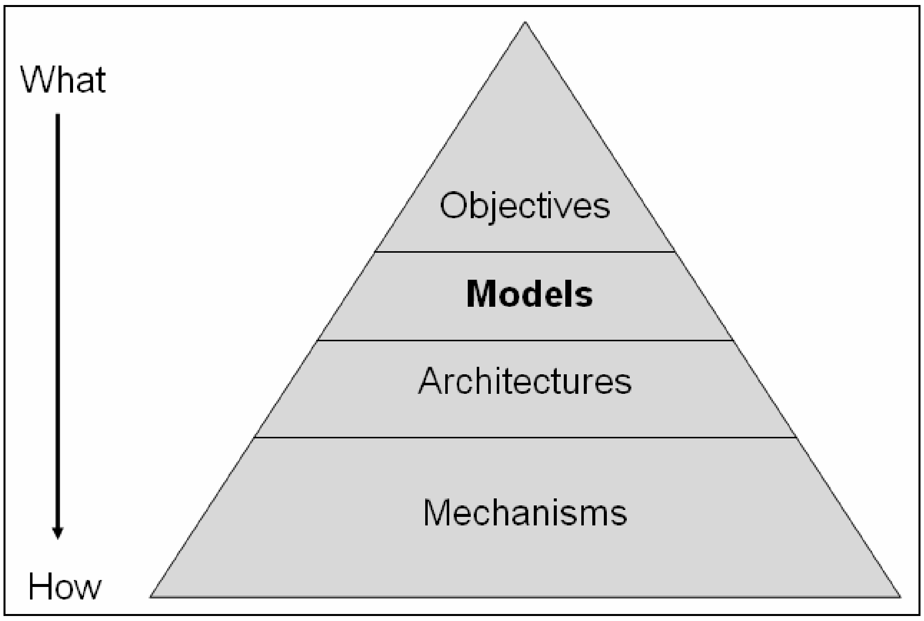


Figure 3: The OM-AM framework for security engineering [45]. Objective and model layers are concerned with articulating what the security objectives are, while architecture and mechanism address how to meet these requirements. The pyramid represents that upper levels are abstract and moving downwards means a refinement process [46].

This work is about developing a suitable access control model when using grids for medical applications. A deep understanding of the neighboring two layers, the objectives and architectures layers is important because both of them are basically implemented. Grid architectures do exist [32, 40, 51-55] and the objectives/policies are well known and studied – for biomedical application at least [56-67]. Less work has been done to develop an access control model for grids, that if any. Grid security implementations consider access control models known and accepted in other environment disregarding the particularity of grid computing itself and the particularity of the architecture layer in this new computing environment. This resulted in using inappropriate models for the grid computing projects. Anderson points out that "many security failures result from environmental change undermining a security model. Mechanisms that were adequate in a restricted environment often fail in a more general one" [1].

2.2.1. Traditional and currently used access control models in closed systems

In literature, authorization is defined as the act of providing and checking the authority of the user (or the job working on her behalf) on a specific set of resources. Access control has a broader meaning; in addition to authorizing users, i.e. answering the question “who can do what”, an access control system may constrain when and how the resource may be used [21, 68]. A deeper and comprehensive discussion about authorization and its meaning in IT standards, grid computing, and as ISO standards is beyond the scope of this work and can be found in literature [68-70].

Access control systems have been developed, evolved, and enhanced since around one-third a century. Most access control issues were identified in the 1960s and the early 1970s. Since then, much of the research about access control systems reworked the basic ideas in new contexts or for new environments, e.g. for object-oriented systems (CORBA) [1]. Practically, the developing of access control models and their implementations were driven by the military needs and the research on operating systems. These two domains shaped access control with its currently known models.

The Lampson’s access control matrix

Traditional access control implementations have remained centered around the access control matrix introduced by Lampson in 1971 [71] (see Figure 4), which was an attempt to formalize an access control model for operating systems. The concept of the access matrix is that a right is granted to a user to access an object in a specific mode, such as read or write (access triples of {user,program,file}). Rights are defined to be static; i.e. they exist whether or not the user practices her rights and accesses the object and they enable repeated access. In other words, time as a dimension of granting the user an access to resources is not considered in the design of the access control matrix. In practice, access control lists (ACLs) or capabilities are used. One can consider ACLs as storing the access control matrix by columns for each object. Capabilities are to store the access control matrix by rows for each user (sometimes known as tickets or certificates) [1, 21, 23, 72, 73]. ACL found its way to mostly all operating systems while capabilities were/are being used for distributed systems.

	Operating system	Accounting program	Patient data	Audit trail
User 1	rwx	rwx	rw	r
User 2	x	x	rw	-
User 3	rx	r	r	r

Figure 4: Example of an access control matrix. r for read, w for write, and x for execute right. A right specifies the kind of access that a subject is allowed to process on an object.

The military access control approach

For military systems, confidentiality of data is the primary concern. The goal in such systems is the prevention of information leakage. For this purpose, Bell and LaPadula developed in 1974 an access control model based on the clearance scheme (of the U.S. military) that restricts flows of classified information (read-down/write-up) [74]. This model has been a very influential step and led to the development of various multilevel security systems (MLS).

The commercial sector access control approach

In the commercial sector, the integrity of data was more important than the confidentiality. Biba introduced in 1977 an integrity model based on the multilevel system similar to that from Bell-LaPadula. Data and subjects are grouped into different levels of integrity; so that users do not corrupt data in higher levels and their data may not be corrupted from lower levels (read-up, write-down) [75]. More important for the commercial-oriented security policies was the Clark and Wilson model for data integrity. Their contribution was to introduce the business practices of separation of duty, and thy proposed an abstract model to enforce the rules, i.e. distinguishing the notion of certification (granting rights) from the notation of enforcement (enforcing the rights) [76]. Brewer and Nash introduced in 1989 the Chinese Wall access control model to capture "conflicts of interest between different parties". The main goal in this model is to prevent a breach of confidentiality by insider knowledge through considering the access history [77], which maybe was the first attempt to integrate the time notion into an access control model.

Categorizing access control models

The different access control models could be categorized into three main sets: Mandatory Access Control (MAC), Discretionary Access Control (DAC), and Non-Discretionary Access Control (Non-DAC) [21, 22, 24, 68, 78-80].

Mandatory Access Control is originally defined in the Trusted Computer System Evaluation Criteria (TCSEC) – the Orange Book – [79] as “a means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e. clearance) of subjects to access information of such sensitivity”. Discretionary Access Control is defined as “a means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject”. A variety of DAC and MAC models appeared to accommodate the diverse range of real-world access control policies and needs. Bell-LaPadula is an example of MAC models, and identity-based access control models (models that implement ACLs) are examples for DAC models.

Role based access control model

MAC models were designed from the beginning to meet the needs of U.S. Department of Defense. For commercial and governmental organizations, DAC was the “standard” access control model till the 1990s. In the 1990s new perspectives came up, MAC and DAC were not sufficient anymore for enterprises within industry, business, and civilian government organizations, where “users do not own the objects and the information for which they are allowed to access” [21, 24].

To solve this problem, Ferraiolo and Kuhn introduced in 1992 the Role Based Access Control model (RBAC) [24, 80] (see Figure 5) and new era of Non-Discretionary Access Control models started [78]. The central notion of RBAC is that permissions are associated with roles, and users are assigned to appropriate roles. RBACs provide therefore a means of naming and describing many-to-many relationships between individuals and rights [24, 81]. This greatly simplifies management of permissions [22]. DAC and MAC models require administrators to translate the organizational authorization policy into permissions on objects, i.e.

each object has a list of access permissions that is granted to various users within an organization. Role-based access control (RBAC) provides better manageability in enterprise environments by allowing permissions to be managed in terms of roles [82]. The difference between RBAC and DAC is that users cannot pass access permissions on to other users at their discretion. While the difference between RBAC and MAC is that MAC policy consider one type of capability: who can read what information, within a role-based system, the concern is the integrity of information: “who can perform what acts on what information” [24].

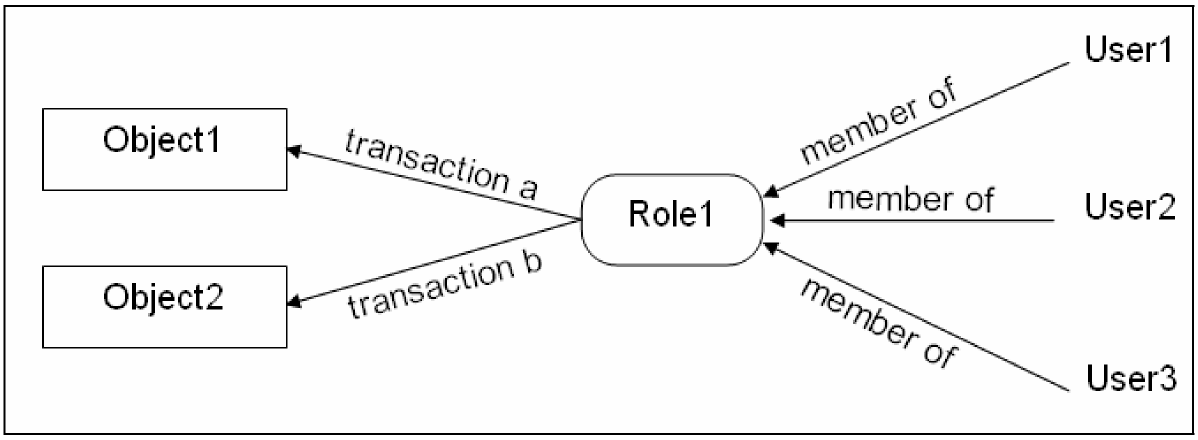


Figure 5: Role relationships. A transaction is a transformation procedure plus a set of associated data items [24, 81]. The orientation in the figure is kept from right to left as in the original paper from Ferraiolo [24].

Other Non-DAC models include task-based [83] and lattice-based access control [84]. In a task-based model the authorization decision is made according to the user’s responsibilities and duties. In a lattice-based model the user should have the greatest lower and the least upper bounds of access rights to the object in order to be authorized.

The RBAC as a Non-DAC model received enormous attention from the security community for its ability to model the real business relationships as well as the different aspects of the other access control models [85, 86]. Sandhu et al. added important extensions to RBAC regarding hierarchies and constrains [22]. A refined model of RBAC [27] has been adopted by the American National Standards Institute, International Committee for Information Technology Standards (ANSI/INCITS) as the ANSI INCITS 359-2004 standard [87]. This standard received a critique in 2005 for its incompleteness to which Ferraiolo, Kuhn, and Sandhu responded, both were published in 2007 [81, 88]. Implementation into operating systems appeared in Windows Server 2003 [82], Red Hat Enterprise Linux 5 (2007) [89], and Solaris 8 (2000) [90].

The RBAC model is defined using Set Theory[3] and has the following basic components [21, 22, 24]:

- *Users, Roles, Permissions,* and *Sessions* are sets of users, roles, permissions, and sessions,
- *PermissionAssignment* $\subseteq$ *Permissions x Roles*, a many-to-many permission to role assignment relation,
- *UserAssignment* $\subseteq$ *Users x Roles*, a many-to-many user to role assignment relation.

RBAC extensions used “sessions” to express that a user has different roles during different accesses, which is an attempt to model time. Nevertheless, Set Theory, as a formal model, cannot capture the notion of time. Therefore, there was an attempt to make RBAC more expressive and to extend it to include the time aspect by introducing temporal logic in the formal model [91]. The central idea is still, anyhow, mapping to roles, which is not what authorization is all about in a modern distributed IT-system. The next section (2.2.2) discusses access control from the distributed system’s point of view. Section 2.3.2 introduces a more modern authorization model than RBAC, i.e. the Usage Control model.

2.2.2. Access control models in distributed systems

Lampson, the introducer of the access control matrix model [71], published with his colleagues in 1993 that in distributed systems the idea of an ACL entry granting access to users according to their identity was no longer as simple as in earlier systems. Traditional access control has focused on the protection of computer and information resources in a closed system environment [92], to which (LAN-based) distributed systems are/were new environments with new requirements regarding access control models.

[3] A short introduction to Set Theory and its relation to some computer science logics are included in 3.3.1.

Authentication-based access control

One effort to solve the authorization problem in the new environment was to provide authentication in distributed systems. Once a remote user is securely authenticated, ACLs on the server-side can be used to provide authorization. Needham and Schroeder protocol and Kerberos are examples for these efforts [93, 94].

Capability-based access control

Anyhow, the early work on an access control model for distributed computing systems goes back to the 1970s as Wulf et al. published the design of the HYDRA operating system [95]. Beside Wulf's work, later works on distributed operating systems like the Cambridge Distributed Computing System [96] and the Tanenbaum's work on Amoeba [97] employed the idea of capability based access control for distributed systems. For the protection of the capabilities in the distributed environment suitable cryptographic mechanisms were used (hashing the capabilities [46][4]). Capability-based models did not provide a real solution. The origin of weakness was that "the right to exercise access carries with it the right to grant access" [98]. Critiques include that such systems cannot detect stolen capabilities, nor prevent duplication of capabilities, and that revocation invalidates all capabilities for the target object [46].

Credential-based access control

The introduction of credential-based systems solved the shortcomings of capability-based approaches. With the notion of credential-based authorization, trust management was born and authorizations for strangers became possible [99-101]. Credential-based systems utilize user's capabilities for authorization in the form of digital credentials or certificates (signed for example by a trusted issuer). Early approaches go back to the late 1980s when Gong modified the semantics of the traditional capabilities to incorporate the user's identity [102][5]. Different approaches enrich the idea of using credential-based access control. Most notably is the Public Key Infrastructure.

[4] *hash = f(secret; protected fields)*, where f is the hashing function, secret is the secret associated with the object, and protected fields can include any information, such as the object identifier and the access rights. And a user's capability would be *capability = (protected fields; hash)*.

[5] *hash = f(secret; principal id; protected fields), certifcate = (protected fields; hash).*

Comments on access control models in distributed systems

Access control models have focused on protecting digital resources on the server-side and do not deal with client-side controls for locally stored digital information. In order to control the usage of already disseminated digital objects, the Digital Right Management (DRM) gave the access control problem a new perspective [103, 104]. DRM technologies attempt to control the use of disseminated digital media by preventing unauthorized access (copy or conversion to other formats) by end users. They have emerged in mid-1990s and gained attention because of their applications in the commercial sector. Current DRM solutions focus on commercial use cases, mainly on intellectual property rights (IPR) protection [25, 105, 106]. Lately, DRM is being addressed for medical applications [107, 108] and general DRM use scenarios include different medical use cases [109].

To sum up, the models for access control in distributed systems address two issues: server-side and client-side control enforcement. They concentrate on protecting digital objects within an environment that consists of a user that interact with a service provider to access the digital object. Third parties (like Certificate Authorities in Public Key Infrastructure) play a passive role in delivering any needed information in a trusted way, i.e. assuring the identity of the user to the service. They do have certificating policy (i.e. an authentication policy), but they do not have an access control policy regarding the users nor can they enforce such a policy. Next subsection (2.2.3) discusses these aspects in a grid computing environment and shows how third parties (authorization authorities) have an active role in defining authentication as well as authorization policies.

2.2.3. Grid computing access control systems

An elementary principle in grids is the Virtual Organization (VO). VO is a dynamic number of individuals and institutions, which have the same interests and/or requirements of using grid resources, such as using the same software or using a large storage capacity for a specific period of time. That VO is built dynamically from different grid entities, which are services provider as well as consumer. This makes VO an essential conception when trying to build an authorization system. Early grid access control systems managed that by granting all users from the same VO the same rights, i.e. Community Authorization System [32, 110]. For a better understanding, a detailed description about how grids establish user authorization is given in this section.

There are three groups of entities in grid computing authorization systems, i.e. subjects (users or jobs in the grid context), resources (objects), and authorization authorities [69]:

Subject: “An entity that can request, receive, own, transfer, present or delegate an electronic authorization as to exercise a certain right (e.g. a person or process)” [69].

Resource: “A component of the system that provides or hosts services and may enforce access to these services based on a set of rules and policies defined by entities that are authoritative for the particular resource. Access to resources may be enforced either by the resource itself or by other entity – a policy enforcement point – that is located between a resource and the requestor and thus protecting the resource from being accessed in an unauthorized fashion” [69].

Authority: “An administrative entity that is capable of and authoritative for issuing, validating and revoking an electronic means of proof (e.g. electronic token or certificate) such that the named subject (holder) of the issued electronic means is authorized to exercise a certain right or assert a certain attribute. Rights may be implicitly or explicitly present in the electronic proof. A set of policies may determine how authorizations are issued, verified, etc. based on the contractual relationships the authority has established” [69].

Authorities in a grid computing environment

In grid computing there are different types of authorities. Attribute Authorities issue assertions that a given subject has one or more attribute/value pairs. Policy and resource authorities issue authorization policies with respect to resources and services offered by these resources. These authorization policies contain assertions that a given subject has a certain right with respect to a given service. The policy authority operates at a higher level than the resource authority, i.e. it may issue access control policy for a whole site or VO; it is the root of trust [69]. Identity authorities (e.g., the Certification Authorities (CAs) of a Public Key

Infrastructure (PKI)) issue certificates that assert a mapping of cryptographic tokens to subject identities. Identity Authorities enable authentication rather than authorization. An authentication process may be used as an input into the authorization process and the identity of a subject becomes another authorization attribute of that subject. Environmental authorities may define things about the resource environment, such as disk usage, or machine load, or about the distributed environment such as the security of the connection or the Internet Protocol (IP) addresses of client and server.

The communications between the different authorization entities follow otherwise the push or the pull sequence (see Figure 6). With the push sequence, the subject first requests an authorization from an authority (e.g., via an authorization server). The authority may or may not honor the subject’s request. It may then issue and return some message or secured message (token or certificate) that acts as a proof of right (Authorization Assertion). Typically such an assertion has a validity time window associated with it. The assertion may subsequently be used by the subject to request a specific service by contacting the resource. The resource will accept or reject the authorization assertion and will report this back to the requesting subject. With the pull sequence, the subject will contact the resource directly with a request. In order to admit or deny the service request, the resource must contact its authorization authority. The authorization authority will perform an authorization decision and return a message that contains the result of an authorization. The resource will subsequently grant or deny the service to the subject by returning a result message [68-70].

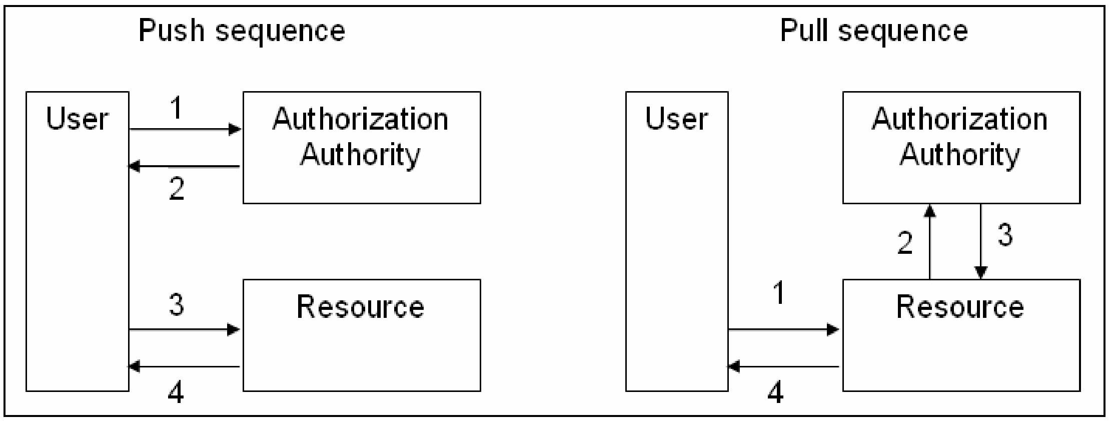


Figure 6: The authorization sequences used in grid computing (Globus Toolkit 4 [111]). The number indicate the sequence order [68-70].

Figure 7 represents an overview of a grid authorization system based on the pull scenario. It also shows various authorities that may be involved in issuing and determining the authorization parameters, attributes and policies. Similar diagrams could be drawn for the push scenario [69].

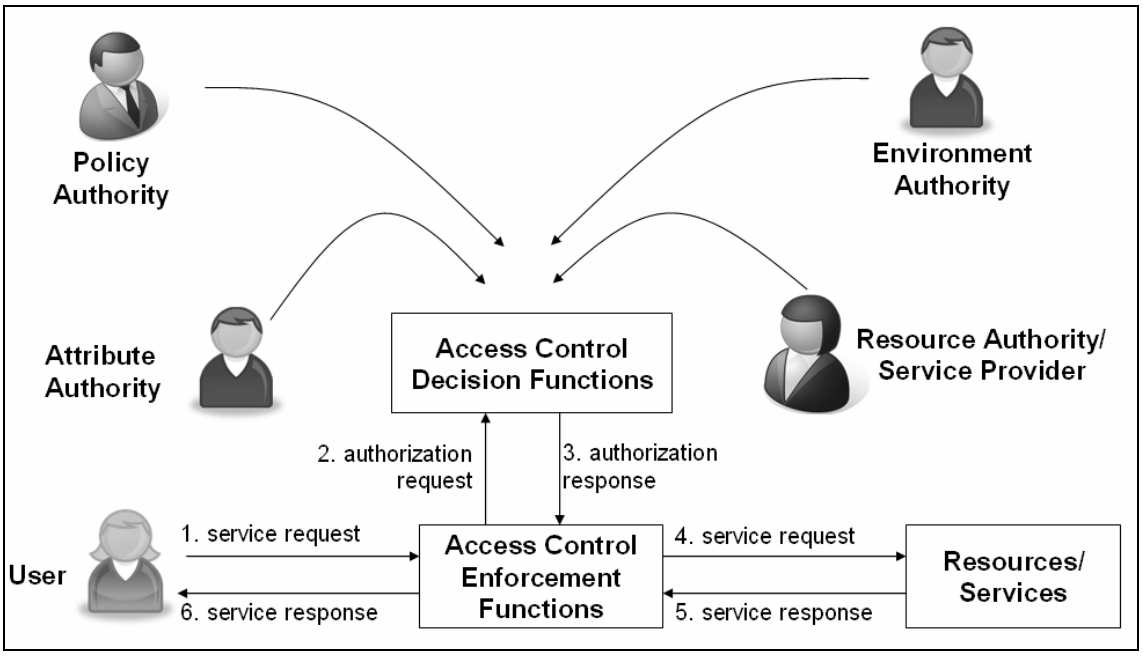


Figure 7: An example of a grid authorization architecture based on the pull sequence (modified from [69]).

The grid extension to the x.509 public key certificate – the proxy certificate

Grid security must address trust between resources with minimal organizational support and allow controlled sharing of resources as well as coordination of resources' sharing. There should be trust establishment possibilities; between resources and VOs, VOs and (new) users as well as between users and resources. Trust should be established in dynamic, distributed user communities [111]. For this purpose, Grid Security Infrastructure (GSI) uses X.509 public key certificates and Secure Socket Layer (SSL) for authentication. The trust model of the certificates allows an entity to trust another organization's Certification Authority (CA) without requiring the rest of its organization to do so or requiring reciprocation by the trusted CA [112] (the Kerberos model, for example, requires that all cross-domain trust be established at the domain level, meaning that organizations have to agree to allow cross-domain authentication [113], which was a reason why it was not chosen for grid authorization [112]).

The x.509 public key certificates and their issuing CAs provide a sufficient authentication infrastructure for persistent entities. In contrast, grids are very dynamic; computations require that several grid resources be used (where each requires mutual authentication); an object (a user or a job) may need to have agents (local or remote) requesting services on her/its behalf. In grids, the possibility to delegate credentials is essential. An important point here is that users may need to delegate privileges specifically to the job and not to the resource as a whole (i.e., other jobs being run by other users on the resource should not share the rights). Moreover, the need to authenticate repeatedly in a short period of time, is very common in grid scenarios when a user is coordinating a number of resources [112]. These requirements led the grid community and developers to extend the standard SSL protocol and standard x.509 certificate to x.509 Proxy Certificate (RFC 3820) [111, 112, 114, 115].

A proxy consists of a new certificate and a private key. The key pair that is used for the proxy may either be regenerated for each proxy or obtained by other means. The new certificate contains the owner's identity, modified slightly to indicate that it is a proxy. The new certificate is signed by the owner, not the CA (see Figure 8). The certificate includes a time notation after which the proxy should no longer be accepted by others [111, 114, 116]. Proxy certificates inherit all, some or none of the permissions of the user whose End Entity Certificate generated the proxy. **Fehler! Verweisquelle konnte nicht gefunden werden.** summarizes the common inheritance models and their attributes.

Table 1: The different proxy types in grid computing (Globus Toolkit) [117]

Proxy Type	Rights Inherited	Notes
Full Proxy	All	Generated by default by the GT4 command grid-proxy-init and other tools
Limited Proxy	All rights except process creation	Delegated by Globus Resource Allocation Manager (GRAM) by default; Created by the command 'grid-proxy-init -limited'
Independent Proxy	None	Not generally in use; Created by command 'grid-proxy-init -independent'
Restricted Proxy	Undefined	Used as a catch-all term to describe proxy certificates with unrecognized policy languages. Not generally in use.

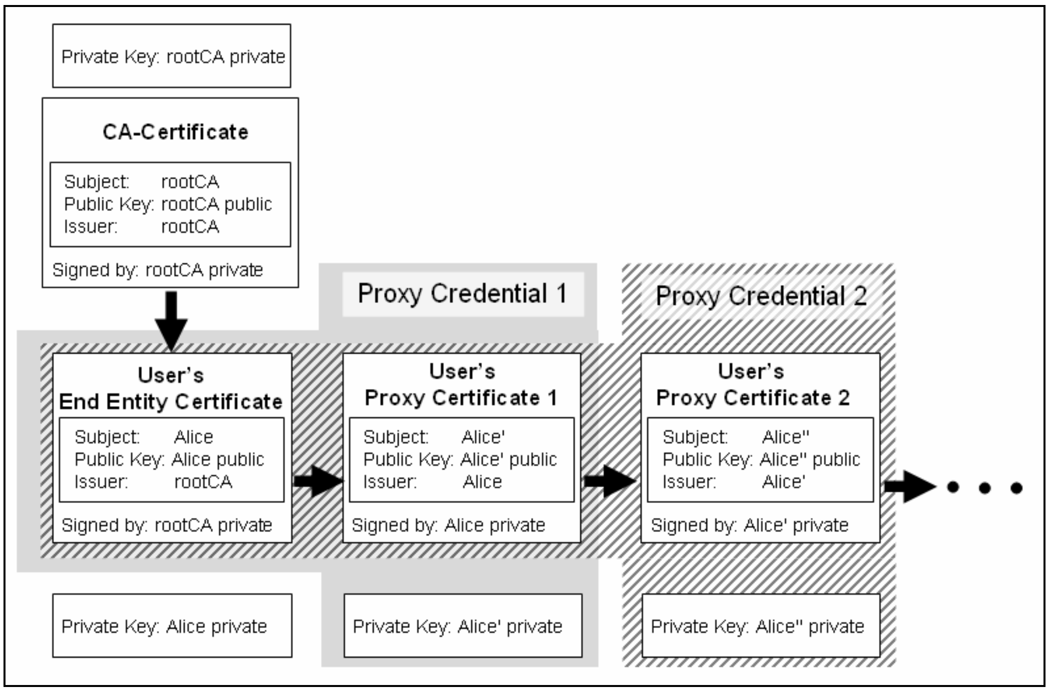

Private Key: rootCA private
CA-Certificate
Subject: rootCA
Public Key: rootCA public
Issuer: rootCA
Signed by: rootCA private
Proxy Credential 1
Proxy Credential 2
User's End Entity Certificate
Subject: Alice
Public Key: Alice public
Issuer: rootCA
Signed by: rootCA private
User's Proxy Certificate 1
Subject: Alice'
Public Key: Alice' public
Issuer: Alice
Signed by: Alice private
User's Proxy Certificate 2
Subject: Alice''
Public Key: Alice'' public
Issuer: Alice'
Signed by: Alice' private
Private Key: Alice private
Private Key: Alice' private
Private Key: Alice'' private

Figure 8: The creation of proxy certificate and credential will establish the trust series. The user will sign only the first proxy [111, 114, 116].

Comments on the proxy certificate

The proxy certificate is one (if not the most important) new protocol presented by the grid computing community, which lay in the middle of the grid computing hourglass architecture model [32, 118] (see Figure 9). The narrow neck of the hourglass defines the set of core abstractions and protocols, onto which many different high-level behaviors can be mapped, and which themselves can be mapped onto many different underlying technologies. Following the grid computing design principle; i.e. to use current protocols and technology and to develop less new protocols, the number of protocols defined at the neck must be small [32].

The main reason for the x.509 proxy certificate is to facilitate delegation of rights in grids. The delegation service can be used when a user wants to delegate rights to a service that is hosted in the same container[6] as the delegation service. The delegation service accepts a credential from the user and provides access to that credential to any authorized service that runs in the same container. Upon

[6] A grid container is a collection of generic primitive services, which are application logic components than can be used by the implemented application, e.g. transaction-, security-, and database management services.

delegation to the service an endpoint reference to the delegated credential is returned to the client, which can then be furnished to other services as a handle to the credential. Detailed description of delegation service, credential creation and refreshment in the grid can be found in the literature [119].

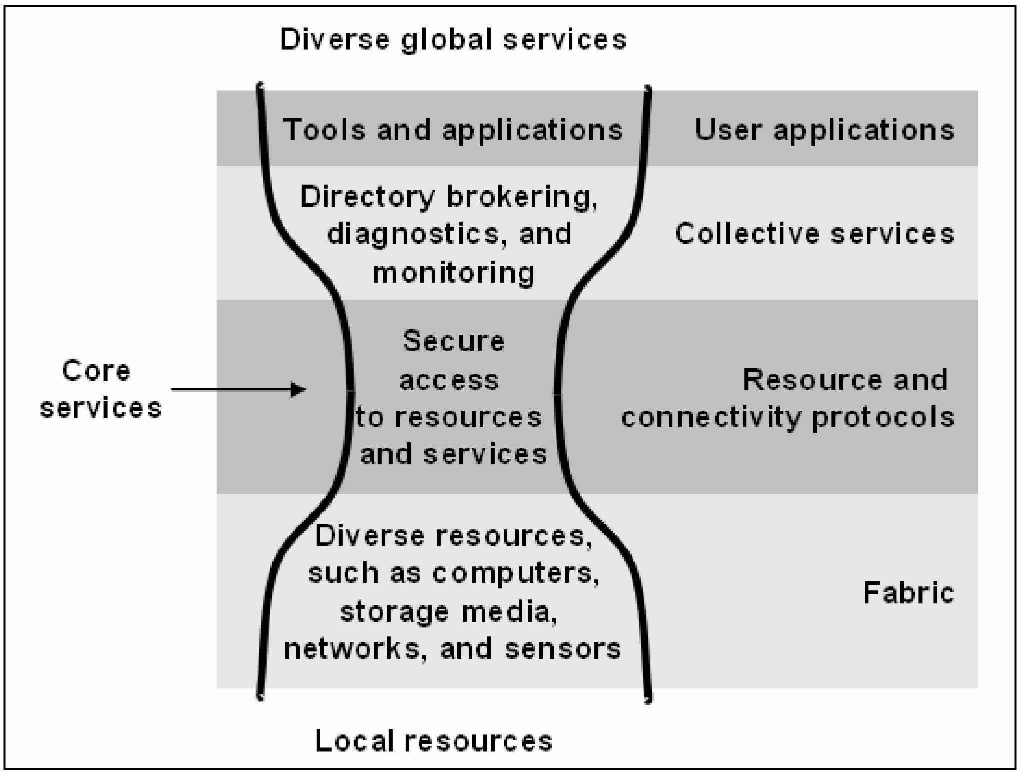

Diverse global services
Tools and applications
User applications
Directory brokering, diagnostics, and monitoring
Collective services
Core services
Secure access to resources and services
Resource and connectivity protocols
Diverse resources, such as computers, storage media, networks, and sensors
Fabric
Local resources

Figure 9: **The hourglass model of the grid architecture. The narrow neck of the hourglass defines the set of core mechanisms and protocols [32].**

Current approaches for authorization in grid computing environments

VOs, different administrative authorities, the trust problem and its solution through delegation and grid proxies are the main new notions and mechanisms of grid computing. There were a lot of efforts to design and implement upon them a suitable authorization framework for grid computing as well as suitable theoretical models to reflect the different visions and requirements of different applications. Three main approaches came from the grid community and were designed from the beginning to be grid authorization systems; these are Grid-Map-File [32], Community Authorization Service (CAS) [32, 110], and Virtual Organization Management Service (VOMS) [120]. Two other systems were designed to be authorization systems in general with the possibility to be used as a plug-in for grid middlewares, these are Akenti [121] and the PrivilEge and Role Management Infrastructure Standards (PERMIS) [122] [7].

Regarding the location of the authorization enforcement process, these approaches can be divided into two categories: virtual organization level authorization systems and resource level authorization systems [68]. Virtual organization level grid authorization systems offer centralized authorization service for an entire VO. Whenever a user wants to access a certain resource, she obtains a credential from the authorization system, which contains certain rights (push sequence). The user presents the credentials to the resource to gain access to it, but the resource holds the final right in allowing or denying the access of the users. CAS and VOMS are VO authorization systems. In contrast to that, resource level authorization systems implement the authorization decision service by the resources themselves. The resources allow users' access according to the credentials presented by the users. This type of authorization systems are Akenti, PERMIS, and Grid-Map-File.

Table 2 summarizes the grid authorization approaches and their characteristics. A detailed discussion and analysis of these different approaches is beyond the aim of this work and can be found in the literature [32, 68, 69, 110, 120-124]. A very good and detailed discussion about these grid authorization systems can be found in [123], [124], and in [68].

For the purpose of discussing grid authorization models in this work, it is important to notice that all mentioned approaches adapt traditional access control models, mainly RBAC and sometimes DAC, with support for the different needed grid specific authorities (see Figure 7) to establish trust. The different visions about how to realize and about the exact role of the authorities are primary reasons why there are differences between these mechanisms in the de-centralizing of decision making (see Table 2).

[7] PRIMA (PRIvilege Management and Authorization in grids), which is developed at Virginia Tech University, does not take the VO concept into consideration. Therefore it is not included here.

Table 2: Comparison between the different grid computing authorization systems (partly from [32, 68, 110, 120-122, 125-127])

	VO based		**Resource based**		
	CAS	VOMS	Akenti	PERMIS	Map-Files
Push/Pull Model	Push	Push	Pull	Push or Pull	Pull
Administrative overhead	Low	Low	Low	Low	High
Authentication	Using GSI	Using GSI	Certificates	Certificates	Using GSI
Revocation	No	No	Possible	Possible	Possible (file-updated)
Interoperability	SAML[8]	SAML	No standard (Complex)	SAML	Minimal
Decision Making	Centralized	Membership centralized Rights de-centralized	Centralized	Centralized	Directly at the resources
Multiple Stakeholders[9]	No	Yes	Yes	No	No
Access control paradigm	All members of a VO have the same rights (can be RBAC if VOs=Roles)	RBAC	RBAC or DAC	RBAC	DAC

2.2.4. Standardization efforts from Health Level Seven regarding access control

The term "Health Level Seven" ("HL7") is used for the organizations involved in developing and supporting the international healthcare standards [128]. HL7's mission is "To provide (global) standards for the exchange, management and integration of data that supports clinical patient care and the management, delivery and evaluation of healthcare services. Specifically, to create flexible, cost effective approaches, standards, guidelines, methodologies, and enable healthcare information system interoperability and sharing of electronic health records" [128, 129].

"Level seven" refers to the highest level of the International Organization for Standardization (ISO) communications model for Open Systems Interconnection

[8] SAML is the Security Assertion Markup Language standard from OASIS.
[9] “Multiple Stakeholders” means the resources are controlled by multiple authorities.

(OSI) - the application level. Within the scope of HL7, this level addresses the definition of the data to be exchanged, the timing of the interchange, and the communication of certain errors to the application. The seventh level supports the security checks, participant identification, availability checks, exchange mechanism negotiations, and data exchange structuring – in other words, it handles access control [128].

Role based access control as the HL7 access control standard

HL7 adapted RBAC as an approach for access control for medical documents and application in 2003 [130]. In May 2007, HL7 balloted the Role-Based Access Control (RBAC) Healthcare Permission Catalogue as a standard and presented a normative language to the permission vocabulary in constructing permissions {operation, object} pairs. In this context, permission is an approval to perform an operation on one or more RBAC protected objects. An operation is an executable image of a program, which upon invocation executes some function for the user. Within a file system, operations might include read, write, and execute. Within a database management system, operations might include insert, delete, append, and update. An object is an entity that contains or receives information. The objects can represent information containers, e.g. files or directories in an operating system, and/or columns, rows, tables, and views within a database management system [131].

Functional and structural roles in the HL7 RBAC

Beside the Sandhu RBAC models adapted by ANSI-RBAC, the hierarchical RBAC and constrained RBAC, HL7 standards define functional and structural roles. Functional roles consist of all the permissions (i.e. operations on health information system objects) needed to perform a task (see Figure 10). A user may be assigned one or more functional roles, and thereby be assigned all of the permissions associated with a corresponding set of healthcare tasks. Permissions will ultimately be used to set the system operations (create, read, update, delete, execute, etc.) for data and software applications.

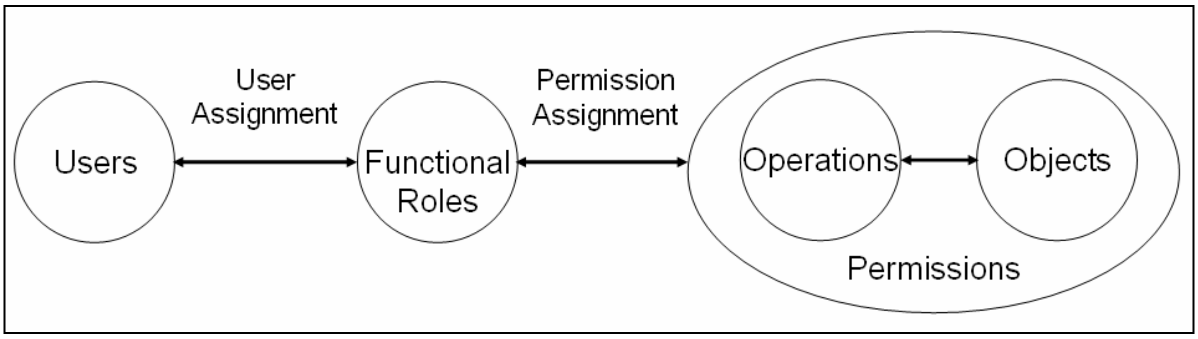


Figure 10: The HL7 role structure is adapted from the ANSI-RBAC core model. ANSI-RBAC describes two other models: Hierarchical RBAC and Constrained RBAC. HL7 does not plan to use these two models in its initial phase (modified from [132]).

Structural roles classify people in the organizational hierarchy as belonging to categories of healthcare personnel warranting differing levels of access control. Structural roles allow users to participate in the organization's workflow (e.g., tasks) by job, title, or position but do not specify detailed permissions on specific information objects. Some structural role examples include: Physician, Pharmacist, and Registered Nurse Supervisor [132].

Structural roles define what specific healthcare workflow users are allowed to participate in while functional roles define authorizations granted to entities to allow access to protected health information. Considering both structural roles and functional roles in the same context, structural roles provide the prerequisites/ competences for entities to perform interactions (or acts) within their specific functional roles [133].

2.3. The time aspect in access control models

The central idea of all traditional access control models was that there is a subject/user who wants to access an object. Access control was mainly about answering the question: who can do what?; LaPadula managed that by answering the question: who has the right clearance to do what?; Context-aware access control concentrated on in which context the access shall be granted, Chinese Wall presented the idea of including the access history in the decision. RBAC reworked the access control scheme by presenting the notion of roles; which role can do what and who has which role.

Different attempts to extend the traditional access control models resulted in a variety of refined models that accommodate the diverse range of the real world access control needs: for example context-aware extensions based on the RBAC model for e-Services [134], context-aware access control for healthcare applications [135], privacy-aware RBAC [136], integrating trust relation into RBAC model [137], dynamic separation of duties in RBAC [91] and many more.

Newly, Sandhu reworked the access control idea itself by arguing that access control should take care of three further questions:

- When and for how long shall the access (or better said "the usage") right be granted? Sandhu represented with this the aspect of time.
- Under which conditions shall the usage right be granted? This question entails the aspects of context-awareness.
- What shall the user as well as the system do before, during and after the usage? This question entails the notion of obligations.

Before giving a short description of the Usage Control model as introduced by Sandhu et al., next subsection (2.3.1) includes a brief introduction into the logic in computer science, transition systems and Temporal Logic, which are essential to understand the Usage Control model and the later chapters of this work.

2.3.1. Predicate Logic and Temporal Logic

Temporal Logic is defined as the logic with the notion of time included [138]. The term Temporal Logic has been used to cover the representation of temporal information within a logical framework. Specifically it has been used to refer to the modal-logic type of approach introduced around 1960 by Arthur Prior under the name of Tense Logic [139] and subsequently developed further by logicians and computer scientists, notably by Amir Pnueli [140]. To better understand Temporal Logics, we can start with the mathematics preliminaries that are well known to the most of us.

Set Theory

Set Theory is the foundation of ordinary mathematics. A set is a collection of elements and is determined completely by its elements. We use the relation $\in$ and we write $x \in S$ to express that the object x is an element of the set S. Operations on sets include $\cap$ (intersection), $\cup$ (union), $\subseteq$ (subset), and $/$ (set difference).

Propositional logic

Algebra is the mathematics of the real numbers and the operators + (addition), - (subtraction), *x* (multiplication), and ÷ (division). Similar to this, propositional logic is the mathematics of the two Boolean values $\top$ verum (true) and $\bot$ falsum (false) and the five operators: $\wedge$ conjunction (and), $\vee$ disjunction (or), $\neg$ negation (not), $\rightarrow$ implication (implies), and $\equiv$ equivalence (is equivalent to). Propositional formulas are defined inductively as follows:
(1) every Boolean variable is a formula, also called atomic formula,
(2) the two values $\top$ and $\bot$ are formulas,
(3) if A, B are formulas, then $A \wedge B$ and $A \vee B$ are formulas,
(4) if A is a formula, then $\neg A$ is a formula,
(5) if A and B are formulas, then $(A \rightarrow B)$ and $(A \equiv B)$ are formulas.

A Boolean value is either true or false. The interpretation for a set of Boolean variables P is a mapping from P to the set of Boolean values *{true, false}*, i.e. Boolean variables are interpreted as Boolean values. Interpretation is also called truth assignment. If we have *I(A) = "true"* for a formula A and interpretation I, we say that A is true in I and we denote that by $I \models A$[10]. If a formula A is true in an interpretation I we say that I satisfies A and that I is a model of A.

A formula A is satisfiable if it is true in some interpretation. A formula A is valid if it is true in every interpretation. Valid formulas are also called tautologies. Two formulas A and B are called equivalent, denoted $A \equiv B$ if every model of A is a model of B, and every model of B is a model of A. Formula evaluation is the decision problem with the answer "yes" if $I \models A$ for the pair (A,I), where A is formula and I is an interpretation.

[10] The symbol $\models$ is the satisfaction relation.

The notions of satisfiability and model can be generalized to sets of formulas. We say that an interpretation I satisfies a set of formulas S, denoted by $I \models S$, if it satisfies every formula in S. If $I \models S$, we also say that I is a model of S. A set of formulas is called satisfiable if there exists an interpretation that satisfies every formula in this set. Automated reasoning deals with automated methods of establishing the satisfiability or unsatisfiability of sets of formulas.

Predicate Logic

Predicate Logic (first-order logic) extends Propositional Logic with two quantifiers: the universal quantification $\forall$ (for all) and the existential quantification $\exists$ (there exists). Quantification is the operation of binding a variable that ranges over a domain of discourse. Hence, the formula $\forall x \in S : F$ asserts that formula F is *true* for every element x in the set S (the domain of discourse). If this is the case, our formula has a logical value of *"true"* otherwise it is *"false"*. For instance, $\forall x \in R : x \geq 0$ is *false* while $\forall x \in N : x \geq 0$ is *true*, where R is the set of real numbers and N is the set of natural numbers.

Propositional Logic and Predicate Logic can be used to describe applications in a static world. "Static" means that the world does not change in time, or that we are interested in one particular state of the world. In most real world applications, we have to reason about systems whose state changes in time, e.g. operating systems, protocols, security protocols, hardware, networks and many others.

Transition systems

A state-changing system [141] is at each particular time moment in a particular state. For the sake of modeling such systems the notions of state and action are introduced. A state is identified by fixed values of all variables at that state. Once a value of a variable (or more) changes, the system moves to a new state (precisely spoken, the system can also stay at the same state). Actions related to the system are responsible of changing the system state by changing the variables' values or when the values are changed by the environment. The actions are carried out by the system itself or can also be performed by the system environment. Sometime we use the word transition instead of action; hence we call such systems “transition systems”.

There are two main types of transition systems, namely reactive and concurrent systems. Reactive systems interact with their environment and change their state in result. Concurrent systems are the systems that consist of multiple, interacting processes or components. The functioning of each of these components can be described independently, but they may also interact with each other through some kind of communication channels, e.g. shared variables.

Formal reasoning

Formal reasoning about transition systems usually follows two steps:
(1) Building a formal model of the transition system which describes the functionalities of the system or an abstraction of these functionalities (e.g. Kripke structure [142])
(2) Using a suitable logic to specify and verify the properties of the system (e.g. Linear Temporal Logic [142]).

The formal model of a transition system could be defined as tuple $S = (X, D, dom, In, T)$ [141], where
(1) X is a finite set of state variables.
(2) D is the domain (a non-empty set). Elements of D are the possible values of the state variables.
(3) *dom* is a mapping from X to the set of non-empty subsets of D. For each state variable $x \in X$, the set $dom(x)$ is called the domain for x.
(4) *In* is a set of initial states of S.
(5) T is a finite set of transitions.
A state of a transition system S is a function $s : X \rightarrow D$ such that for every $x \in X$ we have $s(x) \in dom(x)$. A transition is a set of pairs of states.

Kripke structure

Although the latter definition is very convenient for modeling transition systems, Kripke structures [142] are more convenient for the second step in reasoning about systems; i.e. defining a suitable (temporal) logic. Anyhow, each transition system (in the context of the latter definition) can be converted in a Kripke structure.

Kripke structure assumes a finite set of propositional letters *PL* (in the context of the transition system this will be *PL(X, dom)* and it denotes the set of all interpretations). A Kripke structure is defined as a *tuple K = (S, In, T, L)*, where:

(1) *S* is a finite non-empty of states.

(2) $In \subseteq S$ is a non-empty set of initial states.

(3) $T \subseteq S \times S$ is transition relation that is total, i.e. for each $s \in S$, there is a state *s´* such that *sTs´*.

(4) $L : S \rightarrow 2^{PL}$ is the labeling function of *K*; i.e. a valuation function. (2^{PL} is the set of all sub-set of *PL*).

Linear temporal logic and branching temporal logic

Different logics are used to specify and verify the properties of systems. The most common ones are the Linear Temporal Logic (LTL) and the logic of branching time; i.e. the Computational Tree Logic (CTL).

The set of LTL formulas is the smallest set such that[11]:

(1) each propositional letter $p \in PL$ is a formula,

(2) if φ and ψ are formulas, then so are $\varphi \wedge \psi$ and $\neg\varphi$,

(3) if φ and ψ are formulas, then so are $\bigcirc\varphi$ and $\varphi\mathcal{U}\psi$.

An LTL structure *M* is an infinite sequence S_0S_1… with $S_i \in 2^{PL}$ for all $i \geq 0$. The satisfaction of LTL formulas in *M* at time points $n \in N$ is defined as the followings:

1. $\mathcal{M}, n \vDash p$ iff $p \in S_n$, for all $p \in PL$
2. $\mathcal{M}, n \vDash \varphi \wedge \psi$ iff $\mathcal{M}, n \vDash \varphi$ and $\mathcal{M}, n \vDash \psi$
3. $\mathcal{M}, n \vDash \neg\varphi$ iff $\mathcal{M}, n \nvDash \varphi$
4. $\mathcal{M}, n \vDash \bigcirc\varphi$ iff $\mathcal{M}, n+1 \vDash \varphi$
5. $\mathcal{M}, n \vDash \varphi\mathcal{U}\psi$ iff $\exists m \geq n : \mathcal{M}, m \vDash \varphi \wedge \forall k \in \{n, ..., m-1\} : \mathcal{M}, k \vDash \psi.$

[11] Different authors define LTL in different ways; i.e. including the future *box* and the future *diamond* from the beginning or define them using the *until* and the *next* operator.

The bridge between LTL (logic) and Kripke structures (transition system) is that every computation of a Kripke structure produces an LTL structure. For instance if *K = (S, In, T, L)* is a Kripke structure, a computation $s_0, s_1, \ldots$ of *K* induces an LTL structure $L(s_0), L(s_1), \ldots$ which is called a trace of *K*.

Although the upper LTL definition can model the needed aspect of Linear Temporal Logic, normally other useful abbreviations are also introduced through the future diamond and the future box:

- $\Diamond\varphi := \top\,\mathcal{U}\varphi$, i.e. $\mathcal{M}, n \vDash \Diamond\varphi$ iff $\exists m \geq n : \mathcal{M}, m \vDash \varphi$.
- $\Box\varphi := \neg\Diamond\neg\varphi$, i.e. $\mathcal{M}, n \vDash \Box\varphi$ iff $\forall m \geq n : \mathcal{M}, m \vDash \varphi$.

If a state in a Kripke structure has more than a single successor this induces a “branching” in time. Branching temporal logic (CTL) adds to the LTL operators two temporal path quantifiers, namely: the temporal universal quantifier *A* (for all) and the temporal existential quantifier *E* (there exists).

The set of CTL formulas is the smallest set such that:

(1) each propositional letter $p \in PL$ is a formula,

(2) if φ and ψ are formulas, then so are $\varphi \wedge \psi$ and $\neg\varphi$,

(3) if φ and ψ are formulas, then so are $A\bigcirc\varphi$, $E\bigcirc\varphi$, $A(\varphi\mathcal{U}\psi)$, and $E(\varphi\mathcal{U}\psi)$.

CTL semantic is defined using Kripke structure. Let *K = (S, In, T, L)* be a Kripke structure, the CTL formulas’ satisfaction in *K* at state $s \in S$ is defined as follows:

1. $K, s \vDash p$ iff $p \in L(s)$, for all $p \in PL$
2. $K, s \vDash \neg\varphi$ iff $K, s \nvDash \varphi$
3. $K, s \vDash \varphi \wedge \psi$ iff $K, s \vDash \varphi \wedge \psi$ and $K, s \vDash \psi$
4. $K, s \vDash E\bigcirc\varphi$ iff $\exists t \in S$ with sTt: $K, t \vDash \varphi$
5. $K, s \vDash A\bigcirc\varphi$ iff $\forall t \in S$ with sTt: $K, t \vDash \varphi$
6. $K, s \vDash E(\varphi\mathcal{U}\psi)$ iff $\exists\, sq = s_0 s_1 \ldots$ of K with $s_0 = s$ and
   $\exists\, m > k \geq 0$ with $K, s_m \vDash \psi$ and $K, s_k \vDash \varphi$
7. $K, s \vDash A(\varphi\mathcal{U}\psi)$ iff $\forall\, sq = s_0 s_1 \ldots$ of K with $s_0 = s$ and
   $\exists\, m > k \geq 0$ with $K, s_m \vDash \psi$ and $K, s_k \vDash \varphi$.

The branching logic CTL* includes both LTL and CTL by allowing both LTL operators alone and the CTL path quantifiers to be included in the formulas.

Using transition system and Temporal Logic we can define an authorization model that can capture the notion of time [91, 143-145]. In the next section (2.3.2) we will learn about the Usage Control authorization model approach to do this, which is, in spite of shortcomings, the most complete one among the other models.

2.3.2. The Usage Control authorization model and the implementing of the time aspect

The Usage Control model (UCON) is defined using the OM-AM (Objectives, Models, Architectures, and Mechanisms) framework. At the objective layer of the UCON OM-AM model (see Figure 11), the general subject and object attributes are defined. In the model layer, the conceptual and the formal UCON models are presented as in [25, 144, 145]. In the architecture layer, traditional server-side reference monitor (SRM) or emerging client-side reference monitor (CRM) or combination of them are used to support a UCON system [145]. In the mechanisms layer UCON make use of the existing DRM technologies (e.g. watermarking) or trusted computing technologies (providing mechanisms to support client-side reference monitors). A policy can be specified by XML with some standard approaches such as eXtensible Access Control Markup Language (XACML) for security policies and eXtensible rights Markup Language (XrML) for DRM policies [145].

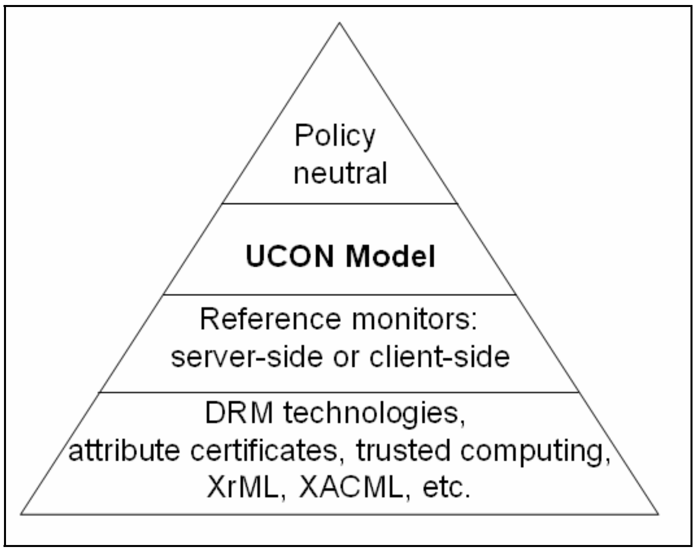

Policy neutral
UCON Model
Reference monitors: server-side or client-side
DRM technologies, attribute certificates, trusted computing, XrML, XACML, etc.

Figure 11: The OM-AM framework for UCON systems as proposed by Sandhu et al. [145]. UCON lays in the model layer and provides abstract interpretation of the requirements.

Usage Control components

The Usage Control system has six components (see Figure 12): subject and attributes, object and attributes, rights, authorizations, obligations, and conditions. The authorization, obligations and conditions are components of usage control decisions. An authorization rule permits or denies access of a subject to an object with a specific right based on subject and object attributes. Obligations are activities that are performed by subjects or by the system. Conditions are system environment restrictions, not related to subject or object attributes.

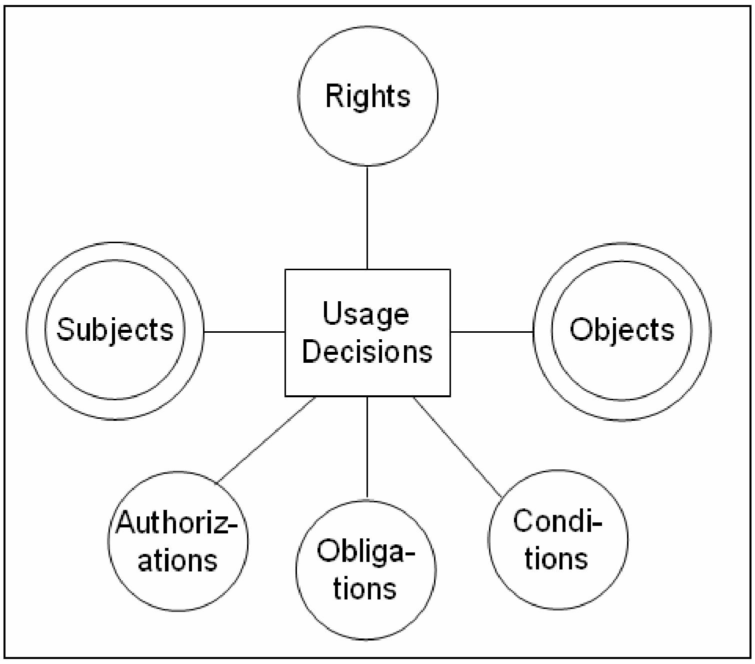


Figure 12: The Usage Control model [23, 144-146]. The intent of the double circles surrounding objects and subjects is to indicate sets, so we have a set of subjects and a set of objects [147].

The phases of a usage process

The usage process consists of three phases: before usage, ongoing usage, and after usage (see Figure 13). To enforce control decisions, UCON distinguishes two different types: pre-decision and ongoing-decision. In the after-usage phase, the original UCON model [23] does not enforce any decision since there is no access control after a user finishes her usage.

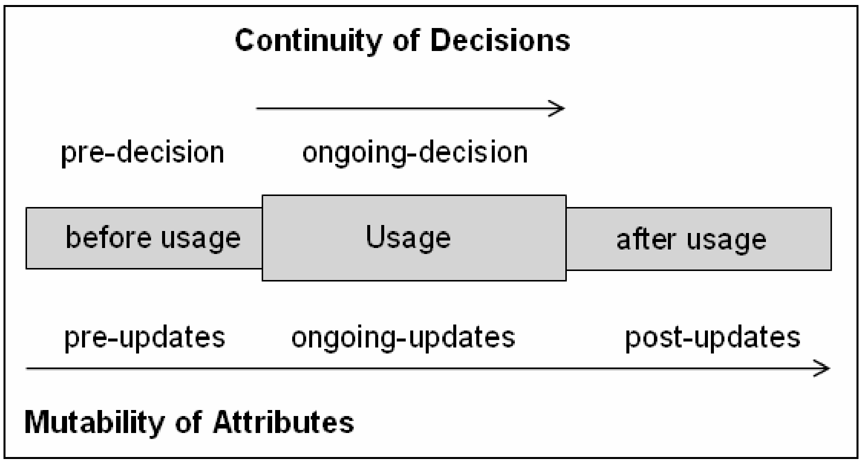


Figure 13: The continuity of decisions and the mutability of attributes in the UCON model. These are the advantages of UCON over the traditional control access models [145].

For mutability of attributes, there are three kinds of updates along the three phases: pre-update, ongoing-update, and post-update. In UCON all these updates are performed and monitored by the system [145], which limits the UCON model in its formal definition to closed systems.

The Usage Control core models

Based on the relationship between authorization decisions and attribute mutability, Park and Sandhu defined seven core models [23, 25]:

a) three pre-authorization models corresponding to a usage decision determined before access and: no subject or object attributes are updated (preA0), attributes are updated before usage (preA1), and attributes are updated after usage (preA3).

b) four ongoing-authorization models corresponding to a usage decision determined during access and: no attributes update (onA0), attributes update before usage (onA1), attributes update during usage (onA2), and attributes update after usage (onA3). In analogy, Park and Sandhu defined the models for oBligations (B) and Conditions (C) like in Table 3.

Table 3: The 16 basic UCON ABC models [23, 25]. “Y” (yes) for included and “N” (no) for not included in UCON models. “A” stands for Authorization, “B” for oBligations, “C” for Conditions

	0 (immutable)	1(pre-update)	2(ongoing-update)	3 (post-update)
preA	Y	Y	N	Y
onA	Y	Y	Y	Y
preB	Y	Y	N	Y
onB	Y	Y	Y	Y
preC	Y	N	N	N
onC	Y	N	N	N

The preA2, preB2, and all mutable condition core models (preC1, preC2, preC3, onC1, onC2, and onC3) are not included in the original UCON core models. Sandhu et al. argued that the reason not to include preA2 and preB2 was that ongoing updates can be postponed to the after-usage phase since the usage decision is already done and will not be affected during the usage process [23, 25]. Zhang, who wrote the UCON specification, extended these models to include all the left possibilities arguing that in a system there may exist concurrent usage processes, i.e. ongoing updates of a usage can affect other usages [145]. For mutable condition core models, subject and/or object attributes can also be updated, for example usage time and usage logs. Similarly, an update in a usage process of condition core models can affect other usage processes [145]. We will see at the end of this section why this is not possible with the UCON formal model.

The Usage Control transition system

In UCON, there are three different kinds of variables: subject attributes, object attributes, and system attributes. The authorization is determined by subjects' attributes, objects' attributes, and rights. A state of a subject or an object is an assignment of values to the attributes. System attributes are variables that are not related to a subject or to an object directly, such as system clock or location. The system has six states[12]; the transition from one state to another is a usage control action (see Figure 14).

The actions are categorized into two classes: actions performed by a subject and actions performed by the system. tryaccess and endaccess are performed by the subject (user) while all other actions are performed by the system. For a usage process, there can be multiple preupdates, onupdate, and postupdate actions for different attributes before, during, and after the access, respectively. Also, in the ongoing usage phase there may be continual or periodical onupdate actions for attributes.

[12] UCON formal model defines a function *state(s, o, r)*, which is a mapping from *{(s, o, r)}* to *{initial, requesting, denied, accessing, revoked, end}*, where *s* stand for subject, *o* for object and *r* for rights.

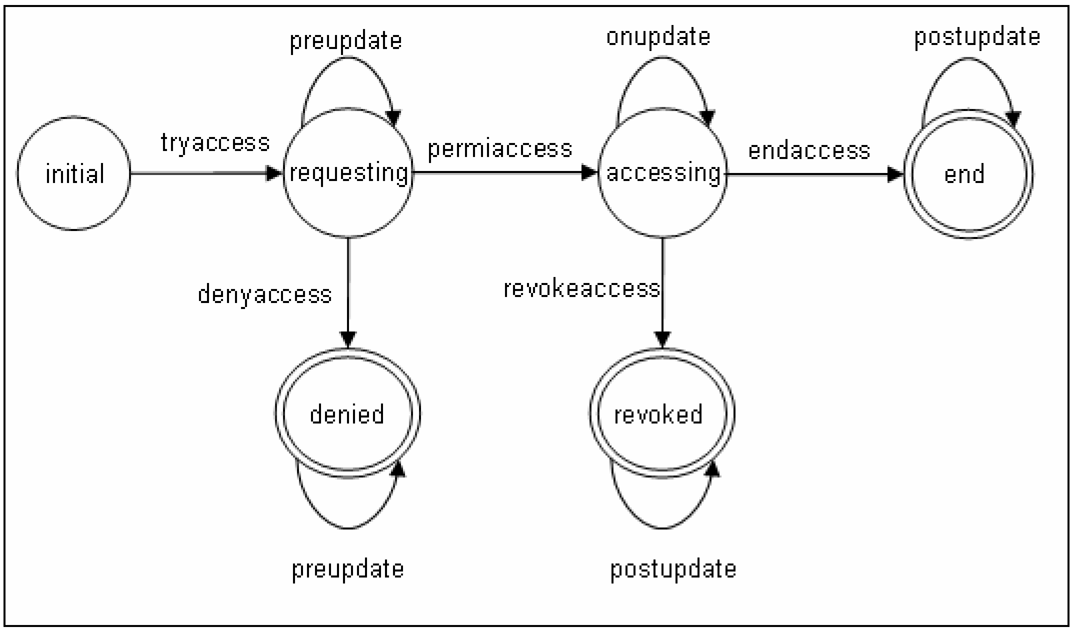


Figure 14: State transition of a single access with Usage Control actions [145]. All actions are performed by one actor, which makes UCON not adequate for grid computing.

The Usage Control formal model

UCON uses Temporal Logic of Action (TLA) [148] with extension to include past actions in order to define the formal model. The formal model uses the basic temporal operators "Always": $\Box$, "Eventually": $\Diamond$, "Next": $\bigcirc$, "Until": $\mathcal{U}$, and the past operator extension: "Has always been": $\blacksquare$, "Once": $\blacklozenge$, "Previous" $\ominus$, and "Since" $\mathcal{S}$. A complete formal definition can be found in the literature [142, 144, 145, 149].

A logical model of UCON is a 5-tuple: $\mathcal{M} = (\mathcal{S}, \mathcal{P}_{\mathcal{A}}, \mathcal{P}_{\mathcal{C}}, \mathcal{A}_{\mathcal{A}}, \mathcal{A}_{\mathcal{B}})$, where $\mathcal{S}$ is a set of sequences of system states, $\mathcal{P}_{\mathcal{A}}$ is a finite set of authorization predicates built from the attributes of subjects and objects, $\mathcal{P}_{\mathcal{C}}$ is a finite set of condition predicates built from the system attributes, $\mathcal{A}_{\mathcal{A}}$ is a finite set of usage control actions, $\mathcal{A}_{\mathcal{B}}$ is a finite set of obligation actions.

A logical formula in UCON is defined by the following grammar: $\phi ::= a|p|(\neg\phi)|(\phi \wedge \phi)|(\phi \to \phi)|\Box\phi|\Diamond\phi|\bigcirc\phi|(\phi\mathcal{U}\phi)|\blacksquare\phi|\blacklozenge\phi|\ominus\phi|(\phi\mathcal{S}\phi)|$,

where *a* is an action, *p* is a predicate.

In a state sequence sq of a model $\mathcal{M}$, the satisfaction relation is defined by induction as followed,

1. $\mathcal{M}, sq, s_0 \vDash p$ iff $s_0[\![p]\!]$, where $p \in \mathcal{P}_{\mathcal{A}} \cup \mathcal{P}_{\mathcal{C}}$

2. $\mathcal{M}, sq, s_0 \vDash a$ iff $s_0[\![a]\!]s_1$, where $a \in \mathcal{A}_{\mathcal{A}} \cup \mathcal{A}_{\mathcal{B}}$ and $s_1$is the next state of s_0 in sq

3. $\mathcal{M}, sq, s_0 \vDash \neg\phi$ iff $\mathcal{M}, sq, s_0 \nvDash \neg\phi$

4. $\mathcal{M}, sq, s_0 \vDash \phi_1 \wedge \phi_2$ iff $\mathcal{M}, sq, s_0 \vDash \phi_1$ and $\mathcal{M}, sq, s_0 \vDash \phi_2$

5. $\mathcal{M}, sq, s_0 \vDash \phi_1 \to \phi_2$ iff $\mathcal{M}, sq, s_0 \nvDash \phi_1$ or $\mathcal{M}, sq, s_0 \vDash \phi_2$

6. $\mathcal{M}, sq, s_0 \vDash \Box\phi$ iff $\forall n \geq 0 : \mathcal{M}, sq, s_n \vDash \phi$

7. $\mathcal{M}, sq, s_0 \vDash \Diamond\phi$ iff $\exists n \geq 0 : \mathcal{M}, sq, s_n \vDash \phi$

8. $\mathcal{M}, sq, s_0 \vDash \bigcirc\phi$ iff $\mathcal{M}, sq, s_1 \vDash \phi$

9. $\mathcal{M}, sq, s_0 \vDash \phi_1 \mathcal{U} \phi_2$ iff $\exists i \geq 0 : \mathcal{M}, sq, s_i \vDash \phi_2 \wedge (0 \leq j < i \to \mathcal{M}, sq, s_j \vDash \phi_1)$

10. $\mathcal{M}, sq, s_0 \vDash \blacksquare\phi$ iff $\forall n < 0 : \mathcal{M}, sq, s_n \vDash \phi$

11. $\mathcal{M}, sq, s_0 \vDash \blacklozenge\phi$ iff $\exists n < 0 : \mathcal{M}, sq, s_n \vDash \phi$

12. $\mathcal{M}, sq, s_0 \vDash \ominus\phi$ iff $\mathcal{M}, sq, s_{-1} \vDash \phi$

13. $\mathcal{M}, sq, s_0 \vDash \phi_1 \mathcal{S} \phi_2$ iff $\exists i < 0 : \mathcal{M}, sq, s_i \vDash \phi_2 \wedge (i < j \leq 0 \to \mathcal{M}, sq, s_j \vDash \phi_1)$.

For example, the usage control policy for the model preA0 is:

$permitaccess(s,o,r) \to \blacklozenge(tryaccess(s,o,r) \wedge (p_1 \wedge ... \wedge p_i)),$

where $p_1, ..., p_i$ are predicates built from subject and/or object attributes, which are preauthorization predicates.

Another example is the usage control policy for the model preA1:

$$permitaccess(s,o,r) \to$$
$$\blacklozenge(tryaccess(s,o,r) \wedge (p_1 \wedge ... \wedge p_i)) \wedge \blacklozenge preupdate(attribute).$$

A detailed description of UCON can be found in the literature [23, 25, 146] as well as the in formal definition [144, 145]. Critiques about the UCON model as well as shortcomings regarding a possible use within a grid environment are included in subsection 4.2.2. Before that, the methods in chapter 3 presents shortly multi-agent systems, the future vision regarding grids and agent systems, game theory, winning strategies and a suitable Temporal Logic to describe these notions.

3. Methods

This work was motivated by studying the emerging problems of transferring a new IT technology – the grid computing – to the medicine discipline. In practice, a variety of theories for technology transfer exist. Anyhow, as grid computing technology itself, and for most its security infrastructure, is still under development, the classical technology transfer theories and approaches [150-152] seem to be inadequate.

As mentioned in the introduction, the purpose of this work is to address data protection and data security for medical application in the grid and to do this precisely in two steps: 1) to identify and address any eventual shortcomings and then 2) to develop a solution proposal to overcome these shortcomings. Both these steps are essential to solve the problems, which appear by transferring the grid computing technology to the medicine discipline.

Identifying the shortcomings of using the grid technology in medicine

The common technology transfer model[13] emphasizes that "the impacts of technology transfer can be understood in terms of <u>who</u> is doing the transfer, <u>how</u> they are doing it, <u>what</u> is being transferred and to <u>whom</u>" [150]. The method used for finding the shortcomings of using the grid technology for the discipline of medicine was primarily by playing an active role in this technology transfer process itself. In other words and in the context of the latter quotation, this is being carried out by:

- merging the "who" and the "whom" questions by working in the MediGRID project (see section 3.1), and
- handling the "how" question by using the knowledge of the three discipline of Grid Computing, Security and Medicine – acquired from the literature (see section 3.2).

[13] The Contingent Effectiveness Model

Developing a solution proposal to overcome the identified shortcomings

After identifying the shortcomings, it was essential to develop a solution proposal for the most urgent shortcomings regarding data protection and data security. Especially for the medicine applications in the grid, it is important to provide a solution for all shortcomings. But this would be far beyond the goals of this work. Nevertheless, such a task can only be handled by collaborative efforts of more than one group of specialists. In this work the intention is to handle the most urgent problems. As the authorization problem in grid computing needs to be tackled urgently, this work focuses on a solution outline for it. Section 3.3 gives a summary of the needed methods for this purpose.

3.1. The D-Grid Initiative and the MediGRID project

The D-Grid Initiative

The origins of D-Grid [153, 154] back to 2003 when the Federal Ministry of Education and Research jointly with the Federal Ministry of Economics and Labor in Germany published a strategic paper [155], which examined the status and consequences of grid technology on the scientific research in Germany and recommended a strategic implementation of this technology. This resulted in the German D-Grid Initiative founded by the German Federal Ministry of Education and Research (BMBF). The funding of the D-Grid Initiative aims to assemble, set-up and operate the grid infrastructure in three stages [154]:

- D-Grid I, 2005-2008: was lunched to design and develop IT services for scientists by the ‘early adopters’ of the scientific computing communities (see Figure 15) in the discipline of high energy physics (HEP Grid), astrophysics (AstroGrid), earth sciences – climate research (C3 Grid), engineering sciences (InGrid), social sciences (TextGrid), as well as medicine and life sciences (MediGRID).
- D-Grid II, 2007-2010: was lunched to design IT services for scientists, industry, and business, including applications in the construction industry, finance, aerospace and automotive, enterprise information and resource planning systems, geographical data, and general IT services.

- D-Grid III, 2009-2011: the strategic goal of the third funding period is to extend the current grid infrastructure with possibilities for Service Level Agreement (SLA) and with a knowledge management layer [41].

Beside the community grid projects, the D-Grid Integration project provides infrastructure components that are shared among the various projects participating in D-Grid. It supports those projects in establishing their own community-specific grid infrastructure. In addition, it integrates those infrastructure components that are developed by a community project and are of interest to grids of other communities.

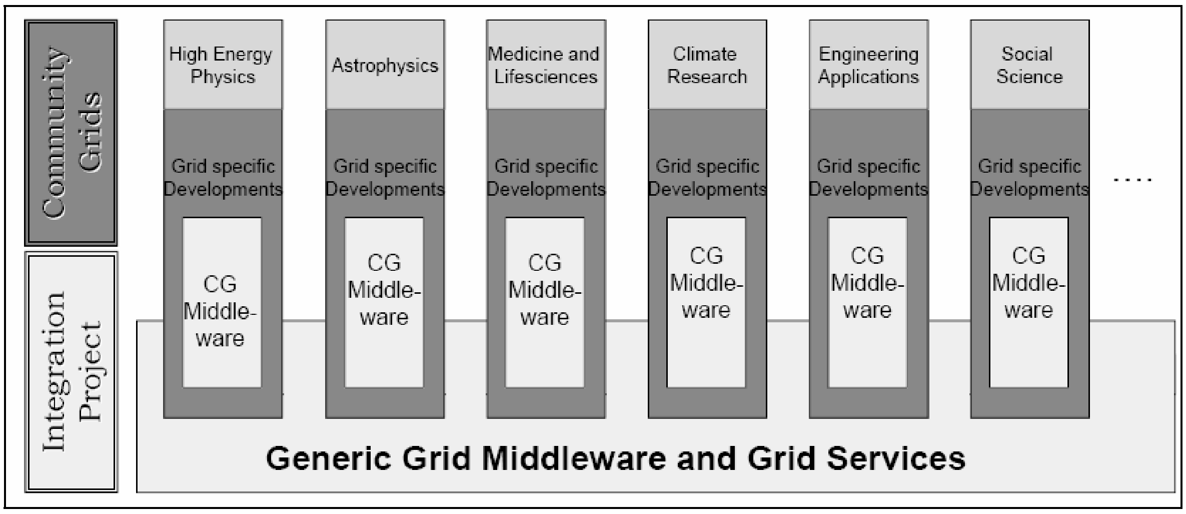


Figure 15: The D-Grid initiative was started in 2005 with funding six projects.

The MediGRID Project

The MediGRID project is hosted at the department of Medical Informatics at the University of Goettingen. The aim of MediGRID is to develop a grid infrastructure for biomedical research and to demonstrate the possibilities of grid computing by implementing key applications in selected domains for medicine and life sciences. These pilot applications can be found in the MediGRID application portal [156]. The MediGRID consortium is organized in eight modules (see Figure 16).

The applications are implemented by the modules image processing, bioinformatics and clinical research. The modules resource fusion and middleware lay the groundwork for the grid computing with resource availability, middleware and workflow management. These major activities are supported by the modules ontology and coordination, which provide support for handling the complexity of medical data and coordinating international activities. The module e-Science (see Figure 16) steer the implementation, perform controlling, and enforce the data security and data protection requirements. As data security is a major concern, the e-Science module is developing policies for users and developers in cooperation with all of the affected parties to make sure that only authorized users can access and manipulate (patient) data. It is also the e-Science module duty – and challenge – to enforce essential process like a complete removing of patient data from the grid when the usage is completed.

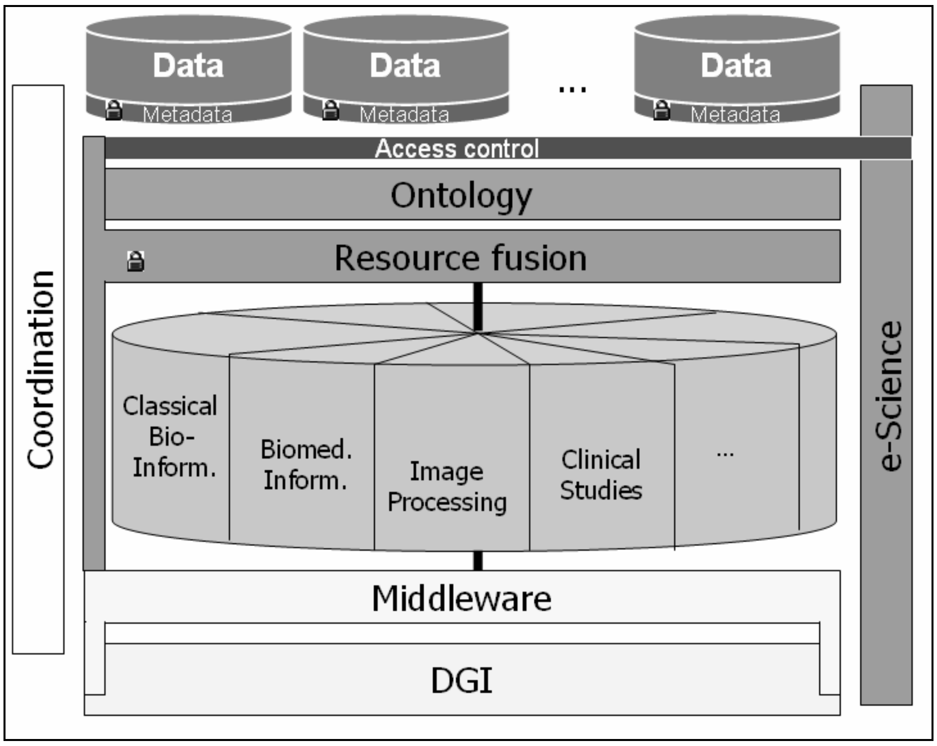


Figure 16: The MediGRID project structure reflecting the different participating modules.

Working as a member of the e-Science module in the MediGRID project opened the possibility to be in contact with the different developers, users, and working groups of the D-Grid project. This cooperation helped to be able to obtain and discuss unpublished and/or unindexed literature (grey literature) about actual and future related data security and data protection topics, including projects related future plans. Moreover, working directly with the InGrid and the DGI projects to develop and implement security standards for the German D-Grid communities was very important: to get to know the different requirements, approaches, solutions, and trends and to try to play a role in steering the plans and observing the acceptance within the different communities. Parts of these work results were also disseminated in the development of the MediGRID Enhanced Security package (see subsection 4.1.2).

Only through the co-work in MediGRID and D-Grid it was possible to identify the authorization as the most urgent problem that face transferring the grid computing technology to the different disciplines including medicine.

3.2. The literature search and used literature

An extensive literature search was conducted using the available international and local databases, journals, and books indexes at either the University of Goettingen Library as well as online. The focus was mainly on information technology and medicine literature. Because of the hasty development in the grid computing technology and to keep informed about the state-of-the-art, the literature searches were periodically completed, at least once per two months. Searches were done using combinations of keywords, such as security, grid computing, distributed computing, data protection, process protection, access control, healthcare, HL7, directives, legal framework. Also project related keywords were used; like Globus toolkit, gLite, GSI, VOMS, PERMIS, XACML, and SAML.

While online available literature was a main source of information for this work, different research reports and results disseminated in international workshops and conferences were also an important source of information and pool of hints to steer this work in all its steps (e.g. [43, 64, 65, 157, 158]).

As security and data protection have a deep historical as well as legal and social background, an extensive review on both was carried out in order to be more capable of anticipating the tendency of the near future development in this sector. Anyhow, because the topic of data security and data protection were discussed since the early days of computing, there exist several reviews on this topic as books and journal papers. These were a very valuable sources and offered a practical way to work selectively and to target the important literature (e.g. [1, 21, 131, 159-162]). In order to identify the future trends, a special focus was kept on the recently published literature of key authors, who did develop and introduce milestones of data security and data protection over the last two decades and who still steer this development till now (e.g. [1, 21-24, 26, 27, 32, 45, 80, 81, 112, 163, 164]).

The literature search resulted in an extensive list of references. This was due to the fact that: 1) this work should cover grid computing, security, and medical application security from three points of view: requirements, implementation and shortcomings, and 2) security and data protection themselves are of long tradition and especial historical development. Therefore, it was necessary to keep the references list rich and multi-sided. Main purposes were to construct an overview and to bring the Ideas from different disciplines together in order to build upon all a vision of how a solution should look like.

Why the conducted literature research and working in MediGRID were Important for this work?

This knowledge gained from the in-depth analysis of the available literature was combined with the gained knowledge from the co-working within the MediGRID project and D-Grid Initiative. This helped for better understanding of the security problem of grid computing used for medical application. In other words, the Ackoff lemma was followed to reach the point of designing a solution for the grid security problem. According to the Ackoff model of wisdom [165], the way to reach the ability to design for the future goes through five stages:

- data: which are symbols,
- information: which are data that are processed to be useful; provides answers to "who", "what", "where", and "when" questions,
- knowledge: is application of data and information and answers the "how" questions,
- understanding: is an appreciation of the "why" questions, and
- wisdom: which encompasses evaluated understanding.

Ackoff argues that the first four categories relate to the past; i.e. they deal with what has been or what is known. Only the fifth category, wisdom deals with the future; it implies vision and design. It was noted by Ackoff (and agreed by different authors) that wisdom requires moving successively through the earlier stages.

In this work, the Ackoff’s lemma to move through the different stages was followed. The understanding and the evaluation of this understanding are included in the analysis in the result chapter. Intending to deal with design for the future, i.e. to provide a future solution outline for a suitable authorization model for medical application in a grid environment requires an understanding of three extra notions, which are included in the next section (3.3).

3.3. How to bring the requirements for a suitable grid authorization model together

Describing the grid as brawn and agent systems as brain, Foster, Jennings and Kesselman proposed in 2004 that development in agent system will be very advantageous for the grid community [166]; “The Grid and agent communities both develop concepts and mechanisms for open distributed systems, albeit from different perspectives. The Grid community has historically focused on “brawn”: infrastructure, tools, and applications for reliable and secure resource sharing within dynamic and geographically distributed virtual organizations. In contrast, the agents community has focused on “brain”: autonomous problem solvers that can act flexibly in uncertain and dynamic environments. Yet as the scale and ambition of both Grid and agent deployments increase, we see a convergence of interests, with agent systems requiring robust infrastructure and Grid systems requiring autonomous, flexible behaviors” [166].

The need to have a reliable grid authorization system is obvious [43, 44, 157, 167]. The grid authorization system should be flexible enough to work in the grid dynamic environment, where entities’ anonymity, different authorities, various policies, and conflicting interests are typical. Such an authorization system should be capable of pursuing flexible and autonomous actions in order to meet the security objectives in the grid environment. Especially in the case of having medical application and patient data in a grid computing environment, the security objectives become more complex. The authorization problem in the grid is rather a “brain” than “brawn”, especially now as the “brawn” is out there and is being used [114, 116, 168, 169].

The approach followed in this work is to consider the authorization procedure as a multi-agent system, i.e. a game between the different entities, who are agents aim to win this game. Each entity has its own interests and therefore can develop its own winning strategy. Simultaneously, the latest developments in authorization models should be considered, namely the UCON model [25] described in chapter 2 as well as subsection 2.3.2.

For better understanding why the approach followed in this work is advantageous, this section describes the used methods, i.e. game theory, multi-agent systems and the Alternating-Time Temporal Logic (ATL), which is a suitable temporal logic for multi-agent systems.

3.3.1. The principles of Game Theory

Game theory was developed mainly by John von Neumann and Oscar Morgenstern in 1940s to study economical behaviors; “The typical problems of economic behavior become strictly identical with the mathematical notions of suitable games of strategy” [170]. Later, the Noble Prize holder John F. Nash (known also from the Oscar winning movie “A Beautiful Mind”) made a fundamental contribution to game theory by introducing the notion of equilibrium strategies [170].

Game theory found its way to different disciplines and applications; from economics and social sciences to robotic and even music [171]. Each discipline defines games differently; anyhow, the principles are the same. Games are a particular sort of conflicts, in which *n* agents or groups (known as players) participate. Games have rules, which define the conditions under which the game begins, the possible moves at each stage of play, and the terms of the payoff. Payoff, or outcome, refers to what happens at the end of a game; for instance in chess, payoff is to declare a winner, and in poker, payoff is an amount of money that is determined according to the game rules by the players’ moves during the course of play. From the player point of view, game theory serves as a mathematical analysis of a conflict of interest, with the intent of finding the optimal choices and strategies that, under given conditions, will lead to the desired payoff [170]. Hence, each game has in its structure four components: players, rules, payoffs, and strategies. Games can be classified according to the features of these components; the most obvious classification is the number of players. Games can also be classified into several types by rules, payoff, or by strategy.

Games classification

We have one person, two persons, and n persons (three persons and more) games. In this context, the decision theory can be considered as a one person game, or a game of a single player against nature. Rules determine the number of options or alternative moves during the play of the game, e.g. two alternatives for each player or three alternatives (e.g. scissors-paper-stone). Regarding payoff, games can be a zero-sum (occasionally constant sum) or non-zero sum. We can also classify games according to the strategies we employ. Each player will attempt to anticipate what other players will do; hence a strategy could be a defensive strategy like minimax (to minimize the maximum loss) or an offensive strategy like maximin (to maximize the minimum gain)[14].

We could have more categories with the different combinations of the last four categories. For instance, games can have sequential play which leads to more complex strategies (rule-strategy classification). Examples are tit-for-tat, i.e. respond kindly or tat-for-tit, i.e. respond unkindly.

The game equilibrium

Traditional applications of game theory attempt to find the equilibria, which are sets of strategies, in which players are unlikely to change their behaviors. For two players zero-sum games, the equilibrium is the saddle point of the upward maximin gain curve and a downward minmax curve (in the payoff matrix this is simultaneously a maximum of row minima and a minimum of column maxima). This is what we call Nash equilibrium.

An example of a game

Two players' games can be introduced using the payoff matrix. The Prisoners' Dilemma is a very good known example for two player non-zero-sum game. The two players (prisoners) can choose between either cooperate with each other or to defect (two alternatives), which correspond respectively to denying or confessing. If one prisoner cooperates, she gets off and the other prisoner is locked up long (e.g. 15 years). If they both cooperate, they will both be locked up for a short period (e.g. 1 year). Anyhow, if they both defect, they will both be locked up for moderate time

[14] Games could have more types. The main message here is that games are classified in categories according to the four components: number of players, rules, payoff, and strategies.

(e.g. 5 years). The game's different outcomes are summarized in the payoff matrix. The Nash equilibrium for this game is the defect-defect situation.

Table 4: The Prisoners' Dilemma payoff matrix. In each cell, the lower number on the left is the first player's payoff, while the upper numbers on the right is the second player's payoff

		Player 2	
		Cooperate (deny)	Defect (confess)
Player 1	Cooperate (deny)	-1 -1	0 -15
	Defect (confess)	-15 0	-5 -5

Game theory can be divided into two branches: combinatorial game theory and classical game theory. Combinatorial game theory covers two player games of perfect knowledge such as chess (each player can anticipate all possible moves of the other player). Combinatorial games have no chance element, and players take turns. In classical game theory, players can bet and/or develop strategies. Here, hidden information and chance elements are main features.

The authorization problem as a game

The authorization problem from the traditional point of view is a one player game, i.e. the system is the one who decides to grant or decline the user access/usage. On the one hand side, authorization models yet are limited to this game, on the other hand side, our needs were indeed limited to these kinds of user-provider environment (see subsection 2.1.2). In this context, the notion of obligations, which appeared newly in authorization models [25, 145, 172], could be considered as a step in capturing some of the game theory aspects. Anyhow, this is a challenge without a suitable formal framework.

In a grid infrastructure, entities do have obligations to fulfill, but they can also accommodate themselves in order to maximize their outcome, and they can develop strategies to choose for example between different solutions (services offered on the grid) with different rules. The authorization problem in the grid computing environment is a multi-player game rather than a one player game.

Recalling Figure 2 and Figure 7, one can see how entities participate differently in the authorization process. In traditional authorization models, the problem is to find out whether the user is allowed to use the resource/service or to perform a special task. In a grid computing environment, the authorization problem is extended to finding out how the user is allowed to use the resource/service or to perform a special task, i.e. developing a strategy for obtaining/granting access/usage rights.

In this work, entities in a grid computing environment are considered as players, which are playing the authorization game. Thinking about the numerous entities having different goals and the complexity of the strategies that can appear in a dynamic grid environment, we can anticipate how challenging the problem is. It is not a Prisoners' Dilemma, in which one can easily find the Nash equilibrium; it is a multi-player game, in which strategies are being developed according to different policies, interests and goals. For this purpose, the next section (3.3.2) describes multi-agent systems, which make use of the principles of game theory to intelligently[15] solve problems in open systems.

3.3.2. The principles of multi-agent systems

A precise definition of agents does not exist [173]. Therefore, one has to adopt a summarizing overview of the different approaches. Agents are "conceptual entities that perceive and act [174] in a proactive or reactive manner [175] within an environment where other agents exist and interact with each other [176] based on shared knowledge of communication and representation [177]" [178] (see Figure 17).

The main idea about agents is that they are autonomous, i.e. capable of acting independently and exhibiting control over their internal state in order to meet their design objectives [179].

[15] Multi-agent systems studies are considered foremost as a branch of artificial intelligence studies.

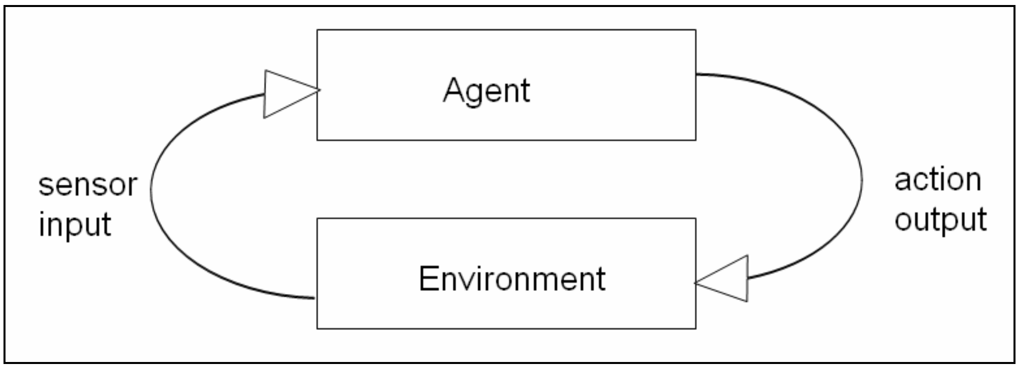


Figure 17: The interaction between an agent and its environment. The agent perceives sensory input from the environment, and produces as output actions that affect it. The interaction is usually an ongoing, non-terminating one [179].

Environments in multi-agent systems

In multi-agent systems, environments are classified according to their properties [179, 180]. Environments can be accessible or inaccessible, where accessible means that agents can obtain complete, accurate, up-to-date information about the environment. Environments can be also deterministic or non-deterministic, where deterministic means that any action has a single guaranteed effect; i.e. no uncertainty about the state that will result from performing the action. When agents have no complete information (e.g. limited sensoric capabilities), they cannot afford complete control and the environment is non-deterministic, i.e. agents' actions can fail. Environments can also be dynamic or static, where static environments change their state only due to the agents' actions, i.e. static environments are predictable. Environments could be also classified in discrete or continuous environments. In discrete environments, the number of states is limited and the environment is always in a specific state.

The most complex environment is an inaccessible, non-deterministic, dynamic and continuous one. Such environment is described as an „open system" [179, 181]. Agents operating in open systems are considered to be intelligent agents [179], which are distinguished with three properties: reactivity, proactivness, and social ability [179, 181]. In order to satisfy their design objectives, reactive[16] agents perceive their environment and respond to its changes, proactive systems can take the initiative to exhibit a goal-oriented behavior, and social agents are capable of interacting with other agents [179]. Similar is the definition of open system from the

[16] Different disciplines define reactivity in different ways.

artificial intelligence point of view. In artificial intelligence, “an open system[17] is one in which the structure of the system itself is capable of dynamically changing. The characteristics of such a system are that its components are not known in advance; can change over time; and can consist of highly heterogeneous agents implemented by different people, at different times, with different software tools and techniques” [182]. In this context and recalling the functionalities of the grid mapping agents (see Figure 2), grid computing has to be considered an open system.

The difference between agents and objects

“Programmers familiar with object-oriented languages such as Java, C++, or Smalltalk fail to see anything novel in the idea of agents” [179]. The difference between agents and objects can be summarized in the following three notions: autonomy, flexibility, and multi-threaded. Agents embody a stronger notion of autonomy than objects, i.e. agents decide for themselves. For instance, one can declare variables or methods in an object-oriented language as *private* or *public*. Using these declarations, the object has control over its state but not over its behavior, i.e. once a method is declared as *public*, the object has no control over whether this method is executed or not. In a multi-agent system, we cannot assure that an agent *i* will perform the action (method) *a* because *j* wants it to, the action *a* may be in a conflict with *i* interest.

Agents can perform flexible behavior (reactive, proactive, and social behaviors). The standard object’s approach does not consider such type of behaviors. Although one can build an object-oriented system to implement such behaviors, the standard object-oriented programming model was not intended to do so.

A multi-agent system is a multi-threaded system, i.e. each agent is assumed to have one thread of control. In the standard object-oriented model, there is a single thread of control in the system.

A mathematical model of multi-agent systems

The mathematical model of multi-agent systems is as the following [179]: we consider a set of environment states $E = \{e, e', ...\}$, a set of agents’ actions $Ac = \{\alpha, \alpha', ...\}$, which transfer the environment from one state to another

[17] here an open system is the environment plus the agents

producing a possible run of an agent in the environment $r: e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_{u-1}} e_u$. We define three sets: *R* the set of all possible runs over *E* and *Ac*; R^{Ac} the set of all possible runs ending with an action; and R^{E} the set of all possible runs ending with an environment state. The state transformer function maps a run to a set of possible environment states, which could results from performing the action $\tau : R^{Ac} \to 2^{E}$. After defining an initial state $e_0 \in E$, we can formally define the environment of a multi-agent system as a triple $Env = \langle E, e_0, \tau \rangle$. Agents are modeled as functions that map runs to actions $Ag : R^{E} \to A$. The definition of multi-agent systems implies that environments are non-deterministic while agents' decision taking is deterministic.

The sequence $(e_0, \alpha_0, e_1, \alpha_1, \ldots,)$ is a run of an agent *Ag* in the environment *Env* if *e₀* is initial state of *Env* and *α₀* = *Ag(e₀)* and $\forall u > 0 : e_u \in \tau((e_0, \alpha_0, \ldots, \alpha_{u-1})) \wedge \alpha_u = Ag((e_0, \alpha_0, \ldots, e_u))$. A discussion about the detailed mathematical model can be found in literature [179].

We can use this mathematical framework to define different agents' behaviors. For example, purely reactive agents can be formally represented by the mapping function $Ag : E \to A$. For example, by considering the control system of a thermostat as an agent, we can model its functionality like the following: $Ag(e) = \text{heater off if } e \geq 20^{\text{O}}\text{C and heater on if } e < 20^{\text{O}}\text{C}$.

Again, following the top-down model and the refinement process, we can refine this multi-agent system abstract model for better usability when building such systems. In order to model the perception behavior of an agent, we split the *Ag* function into two functions: *see* and *action* $Ag = \langle see, action \rangle$ (Figure 18). We define *Per* as the non-empty set of agent percepts. *see* is then the function $see : E \to Per$, which maps the environment states to percepts and *action* is the function $action : Per^{*} \to Ac$, which maps sequences of percepts to actions.

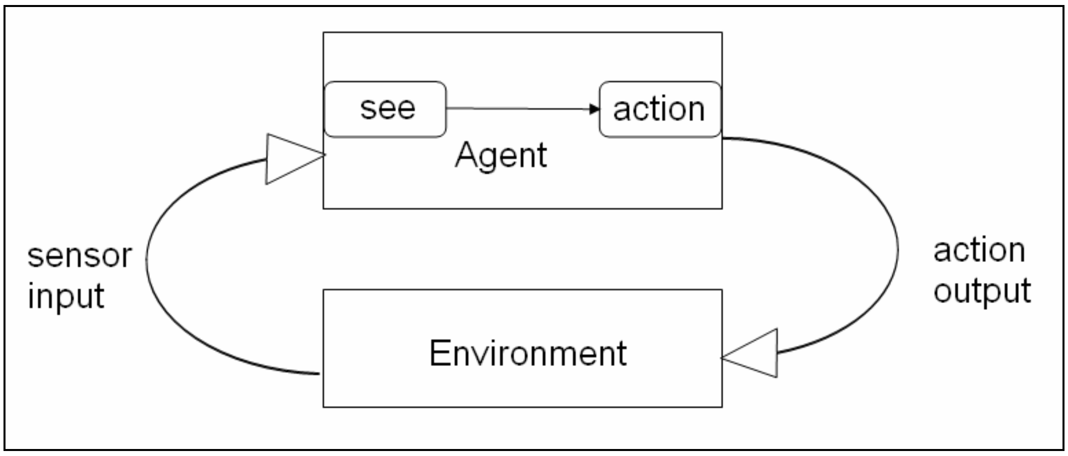


Figure 18: The perception and action modeling in agents systems [179].

Modeling an agent's decision function as a mapping from environment states to action (*Ag* without percept functionality *see*) or from environment states to percepts to actions (*Ag* with percept functionality *see*) allows us to represent the decision functions influenced by history. In order to make agents more flexible in taking decisions, another refinement for the latter model is to consider agents that maintain internal states, which affect the decision. For this purpose we define the set of all internal states of an agent *I*. The *Ag* function becomes $Ag = \langle see, action, next \rangle$, where the *see* function stays like before $see : E \to Per$, the action-selection function would be then $action : I \to Ac$, and the new *next* function represents the mapping from internal state and percept to an internal state $next : I \times Per \to I$.

Designing an agent that behaves correctly is about finding a correct *Ag* function, which captures the targeted objective design. Each agent performs actions using the available information for it. Since all agents are taking actions, each agent must also take the others' actions into account when deciding what to do. Exactly here is the bridge to game theory; i.e. multi-agent systems make use of game theory in their decision making to find the optimal move – the Nash equilibrium. For this purpose, multi-agent systems use a utility function, which is a numeric value reflecting how good the state for the agent is: the higher the utility, the better [179, 183]. Thus, a simple representation of this function is to consider a mapping from each state to a real number representing the utility $u : E \to \mathbb{R}$. Using this, we can define the overall utility of an agent in different ways: a pessimistic strategy of an agent would be to define the utility function according to the worst state the environment could move to, where an optimistic one would be the opposite.

In order to be able to consider long term strategies, we can define the utility function using the runs instead of the individual states. Therefore, the utility function would be $u : R \rightarrow \mathbb{R}$. We denote the probability that a run *r* occurs in the environment *Env* when agent *Ag* in *Env* by *P*(*r*|*Ag*,*Env*). The optimal agent (function) in an environment *Env* is the one that maximizes the expected utility:

$$Ag_{Opt} = \arg\max_{Ag \in AG} \sum_{r \in R(Ag,Env)} u(r) \cdot P(r \mid Ag, Env),$$

where *AG* is the set of all agent functions. As it is easier to speak about the tasks accomplished rather than of a utility value, we can define a binary utility function, i.e. a predicate task specification function $\Psi : R \rightarrow \{0,1\}$. A run *r* satisfies the specification *u* if *u(r)=1*, and fails to satisfy the specification otherwise. We use *Ψ* to denote the predicate specification and *Ψ(r)* to indicate that *r* satisfies *Ψ* [179][18]. All runs *r* of the agent *Ag* in the environment *Env* that satisfy *Ψ* are then: $R_{\Psi}(Ag, Env) = \{r \mid r \in R(Ag, Env) \wedge \Psi(r)\}$. A pessimistic measure of success is when we consider that the agent *Ag* succeeds in the environment *Env* if $R_{\Psi}(Ag, Env) = R(Ag, Env)$, i.e. if every run of *Ag* in *Env* satisfies the specification *Ψ*: $\forall r : r \in R(Ag, Env) \rightarrow \Psi(r)$. An optimistic measure of the agent's success would be to consider whether at least one run of the agent satisfies the specification *Ψ*: $\exists r : r \in R(Ag, Env) \wedge \Psi(r)$. In this context, we can also define the probability that *Ψ* is satisfied by *Ag* in *Env*:

$$P(\Psi \mid Ag, Env) = \sum_{r \in R_{\Psi}(Ag,Env)} P(r \mid Ag, Env).$$

We can classify the environment states into two categories: the goal states *G*, and the states to be avoided *B*. In this context, we can classify tasks into achievement tasks and maintenance tasks. An achievement task is to achieve a goal state $\forall r : r \in R(Ag, Env) \rightarrow (\Psi(r) \leftrightarrow (\exists e : e \in G \wedge e \in r))$, and a maintenance task is to maintain a goal state $\forall r : r \in R(Ag, Env) \rightarrow (\Psi(r) \leftrightarrow (\forall e : e \in r \rightarrow e \in G))$ (which is the same task of avoiding the states in *B*).

[18] This is actually nothing but having a suitable logical framework in order to be able to perform reasoning about decisions in multi-agent systems.

Reasoning about multi-agent systems

We learned in the last chapter that formal reasoning about transition systems usually follows two steps: first we build a formal model of the transition system that describes the functionalities of the system and then we use a suitable logic to specify and verify the properties of the system. The formal model mentioned in this section is to be considered as the first step to have a formal model for multi-agent systems. In the next subsection (3.3.3), we will learn about a suitable temporal logic for reasoning about multi-agent systems; i.e. the Alternating-Time Temporal Logic (ATL).

3.3.3. Alternating-Time Temporal Logic for modeling time and multi-agent systems

Alur et al. introduced in 1997 the Alternating-Time Temporal Logic as the temporal logic for multi-agent systems. An enriched and revised version with changes appeared in 2002. The following description and formal model is taken and summarized from [184].

Summary of Alternating-Time Temporal logic

Alur et al. proposed to enrich temporal logic so that alternating properties can be specified explicitly within the logic instead of using standard Temporal Logic syntax (which was developed for specifying closed systems) and to formulate new semantical conditions for open systems [184]. Alternating-time Temporal Logic (ATL) is interpreted over concurrent game structures. In order to model compositions of open systems, ATL considers a set of players that represent different components of the system and the environment, i.e. the general stochastic game [185]. Classical temporal logics consider only two players games between the system and the environment.

ATL defines a new path quantifier $\langle\langle\ \rangle\rangle$ [184]. For a set $A \subseteq \Sigma$ of players, a set Λ of computations, and a state q of the system, ATL defines a game between a protagonist and an antagonist. The game starts at the state q. At each step and in order to determine the next state, the protagonist resolves the choices controlled by the players in the set *A*, while the antagonist resolves the remaining choices. If the resulting infinite computation belongs to the set Λ, then the protagonist wins;

otherwise, the antagonist wins. If the protagonist has a winning strategy, then the alternating-time formula $\langle\langle A\rangle\rangle\Lambda$ is satisfied in the state q. The path quantifier $\langle\langle A\rangle\rangle$, which is parameterized with the set *A* of players, ranges over all computations that the players in *A* can force the game into, irrespective of how the players in $\Sigma\setminus A$ proceed [184].

The parameterized path quantifier $\langle\langle A\rangle\rangle$ can be considered as a generalization of the path quantifiers of branching-time temporal logics – CTL (see subsection 2.3.1): the existential path quantifier $\exists$ corresponds to $\langle\langle\Sigma\rangle\rangle$, and the universal path quantifier $\forall$ corresponds to $\langle\langle\phi\rangle\rangle$. In particular, Kripke structures can be viewed as game structures with the single player *sys*, which represents the system. The two possible parameterized path quantifiers $\langle\langle\{sys\}\rangle\rangle$ and $\langle\langle\phi\rangle\rangle$ match exactly the path quantifiers $\exists$ and $\forall$ required for specifying closed systems [184].

Depending on the syntax used to specify the set Λ of computations, we obtain two alternating-time temporal logics: ATL and ATL*. In the logic ATL*, the set Λ is specified by a formula of LTL. In the more restricted logic ATL, the set Λ is specified by a single temporal operator applied to a state predicate. By allowing nesting of alternating properties, we obtain ATL as the alternating-time generalization of CTL, and ATL* as the alternating-time generalization of CTL* [184]. We learned that Kripke structure is the natural model for closed systems; similarly, the natural model for compositions of open systems is the concurrent game structure. The latter has the advantage (over Kripke structure) of maintaining the differentiation between the system components and the environment. A concurrent game is played on a state space. In each step of the game, each player chooses a move, and the combination of choices determines a transition from the current state to a successor state. A detailed and formal description of ATL can be found in the literature. The following mathematical foundation of ATL is from [184] and is included here for those interested in the mathematical background of this work.

Concurrent game structure

A concurrent game structure is a tuple $S = \langle k, Q, \Pi, \pi, d, \delta \rangle$ with the following components:

- A natural number $k \geq 1$ of players. We identify the players with the numbers $1,...,k$.
- A finite set Q of states.
- A finite set Π of propositions (also called observables).
- For each state $q \in Q$, a set $\pi(q) \subseteq \Pi$ of propositions true at q, where π is the labeling (or observation) function.
- For each player $a \in \{1,...,k\}$ and each state $q \in Q$, a natural number $d_a(q) \geq 1$ of moves available at state q to the player a. We identify the moves of a player a at the state q with the numbers $1,...,d_a(q)$. For each state $q \in Q$, a move vector at q is a tuple $\langle j_1,...,j_k \rangle$ such that $1 \leq j_a \leq d_a(q)$ for each player a. Given a state $q \in Q$, we write $D(q)$ for the set $(1,...,d_1(q)) \times ... \times (1,...,d_k(q))$ of move vectors. $D(q)$ is called move function.
- For each state $q \in Q$ and each move vector $\langle j_1,...,j_k \rangle \in D(q)$, a state $\delta(q, j_1,...,j_k) \in Q$ that results from state q if each player $a \in \{1,...,k\}$ chooses move j_a. The function δ is called transition function.

For two states q and q', we say that q' is a successor of q if there is a move vector $\langle j_1,...,j_k \rangle \in D(q)$ such that $q' = \delta(q, j_1,...,j_k)$. Thus, q' is a successor of q iff whenever the game is in state q, the players can choose moves so that q' is the next state. A computation of S is an infinite sequence $\lambda = q_0, q_1, q_2, \cdots$ of states such that for all positions $i \geq 0$, the state q_{i+1} is a successor of the state q_i. We refer to a computation starting at state q as a *q*-computation. For a computation λ and a position $i \geq 0$, we use $\lambda[i]$, $\lambda[0,i]$, and $\lambda[i,\infty]$ to denote the ith state of λ, the finite prefix $q_0, q_1, ..., q_i$ of λ, and the infinite suffix $q_i, q_{i+1}, \cdots$ of λ, respectively.

Alternating-Time Temporal Logic Syntax

The temporal logic ATL (Alternating-Time Temporal Logic) is defined with respect to a finite set Π of propositions and a finite set $\Sigma = \{1,...,k\}$ of players. An ATL formula is one of the following:

(S1) p, for propositions $p \in \Pi$.

(S2) $\neg\varphi$ or $\varphi_1 \vee \varphi_2$ where φ, φ_1, and φ_2 are ATL formulas.

(S3) $\langle\langle A\rangle\rangle\bigcirc\varphi$, $\langle\langle A\rangle\rangle\Box\varphi$, or $\langle\langle A\rangle\rangle\varphi_1\mathcal{U}\varphi_2$, where $A\subseteq\Sigma$ is a set of players, and φ, φ_1, and φ_2 are ATL formulas.

The operator $\langle\langle\ \rangle\rangle$ is a path quantifier, and $\bigcirc$ ("next"), $\Box$ ("always"), and $\mathcal{U}$ ("until") are temporal operators. The logic ATL is similar to the branching-time temporal logic CTL, only that path quantifiers are parameterized by sets of players. Sometimes we write $\langle\langle a_1,\dots,a_l\rangle\rangle$ instead of $\langle\langle\{a_1,\dots,a_l\}\rangle\rangle$, and $\langle\langle\ \rangle\rangle$ instead of $\langle\langle\phi\rangle\rangle$. Additional Boolean connectives are defined from $\neg$ and $\vee$ in the usual manner. Similar to CTL, we write $\langle\langle A\rangle\rangle\Diamond\psi$ for $\langle\langle A\rangle\rangle true\mathcal{U}\psi$.

Alternating-Time Temporal Logic Semantics

ATL formulas can be interpreted over the states of a concurrent game structure S that has the same propositions and players. The labeling of the states of S with propositions is used to evaluate the atomic formulas of ATL. The logical connectives: $\neg$ and $\vee$ have the standard interpretation. To evaluate a formula of the form $\langle\langle A\rangle\rangle\psi$ at a state q of S, ATL considers a game between a protagonist and an antagonist. The game proceeds in an infinite sequence of rounds, and after each round, the position of the game is a state of S. The initial position is q.

We consider that the game is in some position u. To update the position, first the protagonist chooses for every player $a\in A$ a move $j_a\in\{1,\dots,d_a(u)\}$. Then, the antagonist chooses for every player $b\in\Sigma/A$ a move $j_b\in\{1,\dots,d_b(u)\}$, and the position of the game is updated to $\delta(u,j_1,\dots,j_k)$. The protagonist wins the game if the resulting computation satisfies the subformula ψ; otherwise, the antagonist wins. The ATL formula $\langle\langle A\rangle\rangle\psi$ is satisfied at the state q iff the protagonist has a winning strategy in this game.

In order to define the semantics of ATL formally, we first define the notion of strategies. Consider a game structure $S=\langle k,Q,\Pi,\pi,d,\delta\rangle$. As before, we write $\Sigma=\{1,\dots,k\}$ for the set of players. A strategy for player $a\in\Sigma$ is a function f_a that maps every nonempty finite state sequence $\lambda\in Q^+$ to a natural number such that if the last state of λ is q, then $f_a(\lambda)\le d_a(q)$. Thus, the strategy f_a determines for every finite prefix λ of a computation a move $f_a(\lambda)$ for player a. Each strategy f_a for player a induces a set of computations that player a can enforce. Given a state $q\in Q$, a set $A\subseteq\{1,\dots,k\}$ of players, and a set $F_A=\{f_a\mid a\in A\}$ of strategies, one for each player in A, we define the outcomes of F_A from q to be the set $out(q, F_A)$ of *q*-computations that the players in A enforce when they follow the strategies in F_A; that is, a computation $\lambda = q_0, q_1, q_2, \dots$ is in $out(q, F_A)$ if $\lambda = q_0, q_1, q_2, \dots$ and for all positions $i \geq 0$, there is a move vector $\langle j_1, \dots, j_k \rangle \in D(q_i)$ such that (1) $j_a = f_a(\lambda[0, i])$ for all players $a \in A$, and (2) $\delta(q_i, j_1, \dots, j_k) = q_{i+1}$.

We can now turn to a formal definition of the semantics of ATL. We write $S, q \vDash \varphi$ to indicate that the state q satisfies the formula φ in the structure S. When S is clear from the context, we omit it and write $q \vDash \varphi$. The satisfaction relation $\vDash$ is defined, for all states q of S, inductively as follows:

- $q \vDash p$ for propositions $p \in \Pi$, iff $p \in \pi(q)$
- $q \vDash \neg\varphi$ iff $q \nvDash \varphi$
- $q \vDash \varphi_1 \vee \varphi_2$ iff $q \vDash \varphi_1$ and $q \vDash \varphi_2$
- $q \vDash \langle\langle A \rangle\rangle \bigcirc \varphi$ iff there exists a set F_A of strategies, one for each player in A, such that for all computations $\lambda \in out(q, F_A)$, we have $\lambda[1] \vDash \varphi$.
- $q \vDash \langle\langle A \rangle\rangle \Box \varphi$ iff there exists a set F_A of strategies, one for each player in A, such that for all computations $\lambda \in out(q, F_A)$ and positions $i \geq 0$, we have $\lambda[1] \vDash \varphi$.
- $q \vDash \langle\langle A \rangle\rangle \varphi_1 \mathcal{U} \varphi_2$ iff there exists a set F_A of strategies, one for each player in A, such that for all computations $\lambda \in out(q, F_A)$ there exists a position $i \geq 0$ such that $\lambda[1] \vDash \varphi_2$ and for all positions $0 \leq j < i$, we have $\lambda[1] \vDash \varphi_1$.

Example of ATL specification

Turn-based synchronous game structure $S_{train} = \langle k, Q, \Pi, \pi, d, \delta \rangle$ shown in Figure 19, which describes a protocol for a train entering a railroad crossing:

- $k = 2$. Player 1 represents the train, and player 2 the gate controller.
- $Q = \{q_0, q_1, q_2, q_3\}$.
- $\Pi = \{out\_of\_gate, in\_gate, request, grant\}$

  $\pi(q_0) = \{out\_of\_gate\}$. The train is outside the gate.

  $\pi(q_1) = \{out\_of\_gate, request\}$. The train is still outside the gate, but has requested to enter.

  $\pi(q_2) = \{out\_of\_gate, grant\}$. The controller has given the train permission to enter the gate.

  $\pi(q_3) = \{in\_gate\}$. The train is in the gate.

- $d_1(q_0) = 2$ and $d_2(q_0) = 1$. At q_0, it is the train's turn. The train can choose to either (move 1) stay outside the gate, in q_0, or (move 2) request to enter the gate and proceed to q_1.

  $d_1(q_1) = 1$ and $d_2(q_1) = 3$. At q_1, it is the controller's turn. The controller can choose to either (move 1) grant the train permission to enter the gate, or (move 2) deny the train's request, or (move 3) delay the handling of the request.

  $d_1(q_2) = 2$ and $d_2(q_2) = 1$. At q_2, it is the train's turn. The train can choose to either (move 1) enter the gate or (move 2) give up its permission to enter the gate.

  $d_1(q_3) = 1$ and $d_2(q_3) = 2$. At q_3, it is the controller's turn. The controller can choose to either (move 1) keep the gate closed or (move 2) reopen the gate to new requests.

  $\delta(q_0,1,1) = q_0$ and $\delta(q_0,2,1) = q_1$.
  $\delta(q_1,1,1) = q_2$ and $\delta(q_1,1,2) = q_0$ and $\delta(q_1,1,3) = q_1$.
  $\delta(q_2,1,1) = q_3$ and $\delta(q_2,2,1) = q_0$.
  $\delta(q_3,1,1) = q_3$ and $\delta(q_3,1,2) = q_0$.

Every state of S_{train} satisfies the following ATL formulas:

- Whenever the train is out of the gate and does not have a grant to enter the gate, the controller can prevent it from entering the gate:
  $\langle\langle\ \rangle\rangle\Box((out\_of\_gate) \rightarrow \langle\langle ctr\rangle\rangle\Box out\_of\_gate)$

- Whenever the train is out of the gate, the controller cannot force it to enter the gate:
  $\langle\langle\ \rangle\rangle\Box(out\_of\_gate) \rightarrow [\![ctr]\!]\Box out\_of\_gate)$[19]

[19] While the ATL formula $\langle\langle A\rangle\rangle\psi$ means that the players in A can cooperate to make ψ true (they can "enforce" ψ), the dual formula $[\![A]\!]\psi$ means that the players in A cannot cooperate to make ψ false (they cannot "avoid" ψ).

- Whenever the train is out of the gate, the train and the controller can cooperate so that the train will enter the gate:
  $\langle\langle\ \rangle\rangle\Box((out\_of\_gate) \rightarrow \langle\langle ctr, train\rangle\rangle\Diamond in\_gate)$

- Whenever the train is out of the gate, it can eventually request a grant for entering the gate, in which case the controller decides whether the grant is given or not:
  $\langle\langle\ \rangle\rangle\Box((out\_of\_gate) \rightarrow \langle\langle train\rangle\rangle\Diamond(request \wedge (\langle\langle ctr\rangle\rangle\Diamond grant) \wedge (\langle\langle ctr\rangle\rangle\Box\neg grant)))$

- Whenever the train is in the gate, the controller can force it out in the next step:
  $\langle\langle\ \rangle\rangle\Box(in\_gate \rightarrow \langle\langle ctr\rangle\rangle\bigcirc out\_of\_gate)$

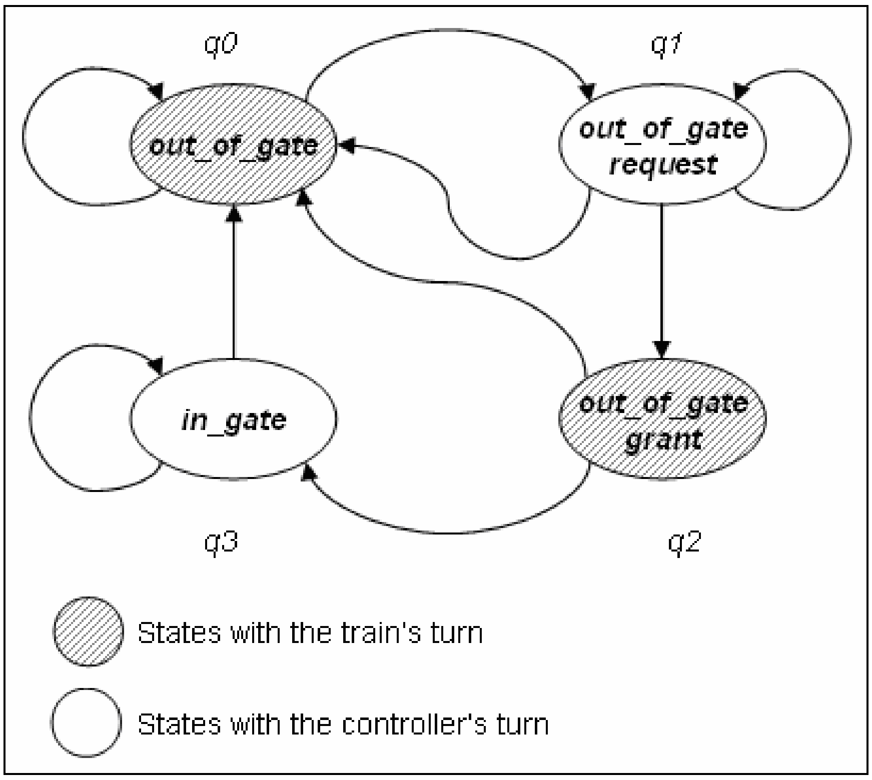

q0
q1
out_of_gate
out_of_gate
request
in_gate
out_of_gate
grant
q3
q2
States with the train's turn
States with the controller's turn

Figure 19: A turn-based synchronous game structure modeling a train controller (adapted from [184]). There is no standard graphical representation method used with concurrent game structures.

Using ATL for modeling authorization in grid computing environment

As mentioned before, the approach followed in this work is to consider the authorization procedure as a multi-agent system, i.e. a game between the different entities, who are agents aim to win this game. In next chapter (4), we are going to learn how each entity has its own interests and own strategy to fulfill its objectives.

While this is not the aim of authorization, it is a very important point for grid computing. Grid computing cannot function without this notion. ATL as a modeling logic allows us to model security, time, and multi-agent in open systems.

4. Results

4.1. Analysis of the grid computing security problem

The first part of this section (4.1.1) elaborates the shortcomings of the current data security and data protection legal framework regarding the grid technology as the performed analysis. In the second part (4.1.2), the shortcomings of the current grid technology, namely the Globus toolkit 4 middleware, regarding the data security and data protection requirements will be presented. The main goal of this section is to give a brief overview of the current security and data protection situation regarding the use of grid computing for (bio)medical application. This brief analysis will show why the authorization problem is the most urgent one among others.

4.1.1. Shortcomings of the (EU) legal framework

Because of the sensitivity of medical data, medical applications should adhere to a strict legal framework, which is at present not ready to deal with the grid aspects. This is true not only for medicine; the similar problem exists also in the grid application in the business and engineering disciplines. There is a legal "grey zone" regarding the concept of grid computing; namely virtualization – whether organizational, i.e. Virtual Organization, or technical, i.e. Virtual Machine. Virtualization is a key technology and fundamental concept of grid computing, still anyhow legally difficult to be handled [186-190].

The concept of Virtual Organizations

A Virtual Organization (VO) is a dynamic number of individuals and institutions, which have the same interests and/or requirements of using the grid resources, such as using the same software or using a large storage capacity for a specific period of time. The interested parties align themselves together in a sort of consortium to achieve certain tasks, which cannot be accomplished by using only their own resources alone [32, 187]. As normally there is no written contract between the parties, this leads to legal difficulties, especially regarding accountability.

For example, if a physician were to perform a virtual surgery simulation in a grid environment, nobody would be able to guarantee the reliability of the calculations in time and the correctness of the results. The entries, the choice of method, the method itself, the data transfer and the reliability of resources against errors or mistakes are all responsibilities, which are shared – without any rules yet – between different people within a VO. A disband of VOs or a possible change of a specific resource provider inside a VO creates more difficult cases concerning accountability.

A similar problem of shared responsibilities within the business sector since the 1990s is the regulation of the so called Virtual Corporation (VC) or Virtual Enterprise. VC stands for cooperation between legally independent companies, institutions and/or individuals, which is based on a common business understanding. The entities participate in a horizontal and vertical cooperation with their core competence and appear to third parties as a single company. The VC stays in existence until it fulfills its purpose [191]. Because of the business nature of the VCs, the participating parties can contractually define their own policy. If they did not, there would possibly be legal problems [192, 193]. In the health sector, and because of the strict legal framework, such a contractual solution is not possible. From the medical care point of view, the service should be failsafe and constantly available. The grid computing environment has to offer a sustainable service. An adjustment of the VOs regulations and management as well as of the legal framework is required before any deployment and usage of medical applications in the grid [64, 188, 189].

The concept of Virtual Machines

A Virtual Machine (VM) is a virtualization at the hardware level or at the software level. In general, a VM provides a virtual system (a guest system) of a specific operating system or a virtual environment for certain software within a host system. Originally it is defined as “an efficient, isolated duplicate of a real machine”. Current use implies no direct correspondence to any real hardware [194, 195]. This results in that the applications can run independent of the real platform and/or architecture.

The Virtual Machine, like VOs, is a key concept of grid computing [196-198], still anyhow a legal challenge. Accountability in case of an error is not completely visible. If a simulation of a therapy was to be carried out between VMs of six physical machines in four institutes and an error occurred, which led to an irreversible time and data loss, it is difficult to determine where and what the mistake exactly was, more difficult would be find out why and who is responsible for that mistake [64]. This is mainly because VM logs and audit files are manipulable forms and are completely dependable on the host system[20]. Moreover, VMs themselves are not error free [199-201].

Virtualization using grid computing

Grid computing opens new opportunities to create a virtual record; i.e. to connect data from different locations and resources. Through the distributed storage and processing of data, grids offer also the possibilities of handling large datasets and performing complex analysis and data mining on the various records as well as real-time data updates. In this context, a true problem in the use of personal data (in medical applications) is the possibility of unauthorized re-identification of individuals, known as the disclosure risk.

Anonymization and pseudonymization in their common sense and methodology are not sufficient in a grid computing environment. Grid computing opens up various opportunities for the re-identification. A high "disclosure risk" is typical for a grid environment [28, 29, 202]. It is no longer the question how secure the environment is, but rather how "disclosive" grid computing is [203]. In February 2004 there was a correspondence between the “National Immunization Program” of the “Centers for Disease Control” and the “Institutional Review Board” of “Northern California Kaiser Permanente” regarding the suspect that two researchers may have harmed the patients’ secrecy by correlating records from different studies. From the letter: ”By attempting to merge data files, the researchers would have created a more complete medical records on subjects, and if so, could have increased the risk of a breach of confidentiality.” This had an inconvenient impact: the data access was blocked for both researchers until a conclusive declaration was presented [64]. Grid computing is the ideal platform for such incidents and its security services still cannot handle such problems.

[20] Dependable is in the sense of having no real physical hardware. For example, host systems can manipulate the clock and the real available memory, they do have access to and control over the data in the memory, etc.

Perspectives on the future legal developments regarding the use grid computing technology

Over the last few decades, constant medical and/or technical developments ensue growing complexity in the relevant legal aspects. A study on the legal aspects of telemedicine from the year 2000 shows how slow the legal developments are. While technical development in current information technology happens very quickly, it is natural that the legal framework is not adequately adjusted (see Figure 20) [159, 204]. Medical and technical innovations can be divided into phases, each of which requires new legal considerations. Usually, there is a lapse of time of several years before an effective regulation of a given legal, technical, and social environment is found. The development of legal regulations moves from a flexible solution, e.g. codes and contracts, to mandatory solutions as laws and regulations. These developments rarely progress in continuous manner and often contain discontinuities [159]. This applies to the grid technology also, which means that a long legal interpretation process is ahead.

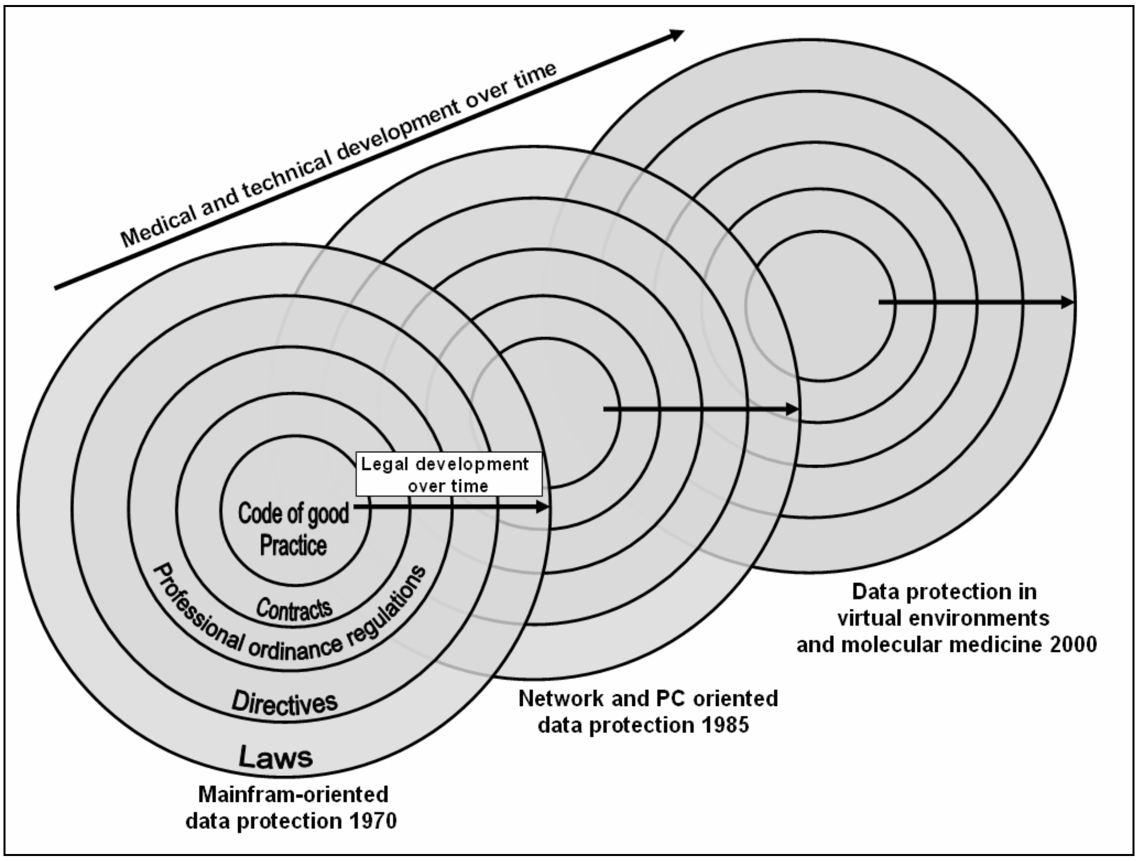

Medical and technical development over time
Legal development over time
Code of good Practice
Contracts
Professional ordinance regulations
Directives
Laws
Mainfram-oriented data protection 1970
Network and PC oriented data protection 1985
Data protection in virtual environments and molecular medicine 2000

Figure 20: The circles stand for an attempt to graphically represent the metamodel using the example of data protection in health care [159, 204].

Virtualization has been discussed since the mid-1990s as a technical solution. But until now there is no legal framework for it. Therefore, it is necessary to study data protection aspects in virtual environments and to develop generic solutions for it. The current technology should be conformed according to the current legal framework [57, 205]. Anyhow, as long as there is no clear legal definition of virtualization concepts, it remains difficult to conform the grid computing technology to a legal framework. A kind of converging process between the legal and the technical developments regarding data protection and data security is necessary (see Figure 21). Similar to how a legal framework was required in order to adapt the development of a public key infrastructure in the past (1999: EU Directive 1999/93 [206] and 2001: SigG in Germany [207]), grids for medical applications will likely require new legal regulations.

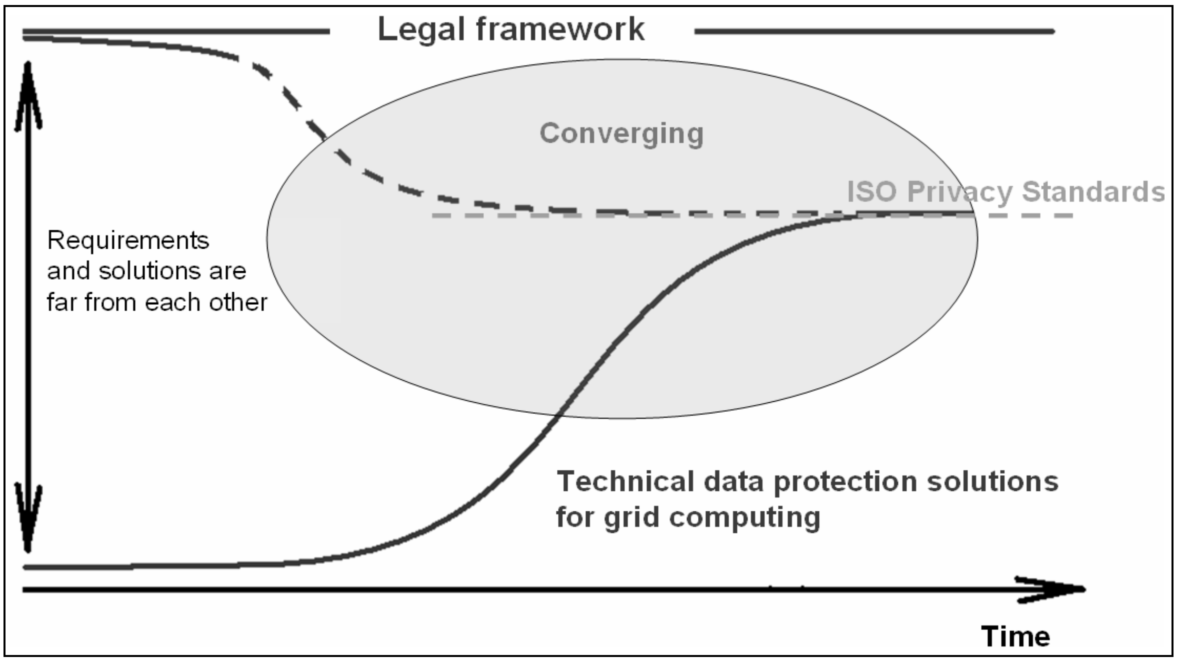

Legal framework
Converging
ISO Privacy Standards
Requirements and solutions are far from each other
Technical data protection solutions for grid computing
Time

Figure 21: The converging between the (legal) requirements and the (technical) solutions is necessary [64].

The 26th international conference on privacy and data protection in Wrocław 2004 resulted in a resolution about a „Privacy Framework Standard“. The resolution urges the International Standards Organization (ISO) to work on privacy and data protection standards: „Development from Privacy Law into Privacy Standards”. The “Privacy Enhancing Technologies” (PET) [208, 209] are of interest for the future ISO Privacy Standard [210]. This development has to be closely monitored in the interest of the biomedical grid community in order to set up a sustainable grid infrastructure.

Each change in the legal framework or in the technology in regard to grid computing use by the biomedical community should take these standards into account. A converging between the legal framework and the technical solutions of data protection and data security to the common ISO privacy standards should be considered [64]. As it is not expected to have these standards before end of 2008 [210], we need to keep track of the development of the ISO privacy standards in order to reduce the converging period later.

4.1.2. Shortcomings of the grid technology in the medical sector

The requirements

A legal framework for the protection, security and transport of personal data as well as patient data is introduced in different legislative guides, directives, or laws (in the EU: 95/46/EC processing of personal data; 97/66/EC protection of privacy in the telecommunications sector; 99/93/EC a framework for electronic signatures; 2002/58/EC privacy and electronic communications). The International Organization for Standardization (ISO) has defined the common security services found in modern IT systems as well. The list was first put in ISO 7498-2 (OSI Security Architecture) and later updated in ISO 10181 (OSI Security Frameworks). Although the implementations vary among countries, they imply the same fundamental requirements regarding data security [62, 211, 212], which are:

- Confidentiality: assurance that data are not made available or disclosed to an unauthorized person.
- Integrity: assurance that data cannot be changed/deleted/altered by an unauthorized party/person.
- Authenticity: assurance that the person is the one she claimed to be.
- Accessibility: upon demand, (patient) data can be accessed and used by authorized people.
- Accountability: assurance that actions of a person, especially modifications that she performs on data, can be traced.

Biomedical data have their special nature. They are not only heterogeneous, but rather they contain different information types and different levels of privacy. They vary from aggregated data describing population and diseases (epidemiology, clinical practice, clinical trials), to more granular patient data and pathological descriptions (health record, clinical history, physical exams) and to cellular and molecular data (histology, genetic test results and genomic data) [3, 63, 213]. Given semantic data interoperability, the researcher can correlate and analyze the data using the suitable biomedical informatics methods and tools. Having these data online with the suitable tools to perform such operations creates new challenges for data protection and data security [64]. In this context additional data protection requirements come up when dealing with person related data [58, 161, 212, 214-216]:

- Data necessity principle: upon demand all person related data of a patient may be disclosed, but not more than the needed data (for the treatment or the research) [58, 161, 215, 216].
- Context of treatment: person related data of a patient should be disclosed only to the personnel participating in her treatment (e.g. in Germany §203 StGB [214])
- Patient consent: the patient should formally agree on the handling of her person related data [58].
- The guarantee of patient rights: the possibility of rectification, blocking, and deletion of her personal data should be offered and guaranteed [58].

In the light of these requirements of data security and data protection, the following analysis of the grid middleware was carried out.

Analysis of the grid middleware

The Grid Security Infrastructure (GSI) in Globus Toolkit version 4 (GT4) was examined to find out whether and how it fulfills the requirements in a service chain in a common healthgrid (see Figure 22 [65]). The Globus Toolkit 4 (GT4) is widely used and is considered as the standard grid middleware. The common service chain in grid for life sciences was inspired from the different healthgrid projects and reflects the usual use of grid computing to build services for the (bio)medicine community.

The security tools in GT4 deal with [116]: authentication (establishing the identity of users or services), communication security, authorization (determining who is allowed to perform what actions), and other supporting functions such as managing user credentials and maintaining group membership information. The newer versions of GT; version 3 and 4, provide web services (WS) authentication and authorization capabilities beside the pre-WS mechanisms [32, 111, 114, 217]. Both use the standard X.509 certificates and proxy certificates [115] to identify persistent entities such as users and servers and to support the temporary delegation of privileges to other entities. A detailed description of the grid use of X.509 and proxy certificates is included in the primer (2.2.3 Grid computing).

The Globus design model intends to use current internet technologies with as less modifications as possible, adapting an “hour glass” model for new standards [114], i.e. write as less new standards as possible and keep it focused on bridging between the existing standards (see Figure 9). In this context, the Globus Security Team implements security as a “five layers grid security infrastructure (GSI)” based on standard X.509 certificates (see Table 5) [32].

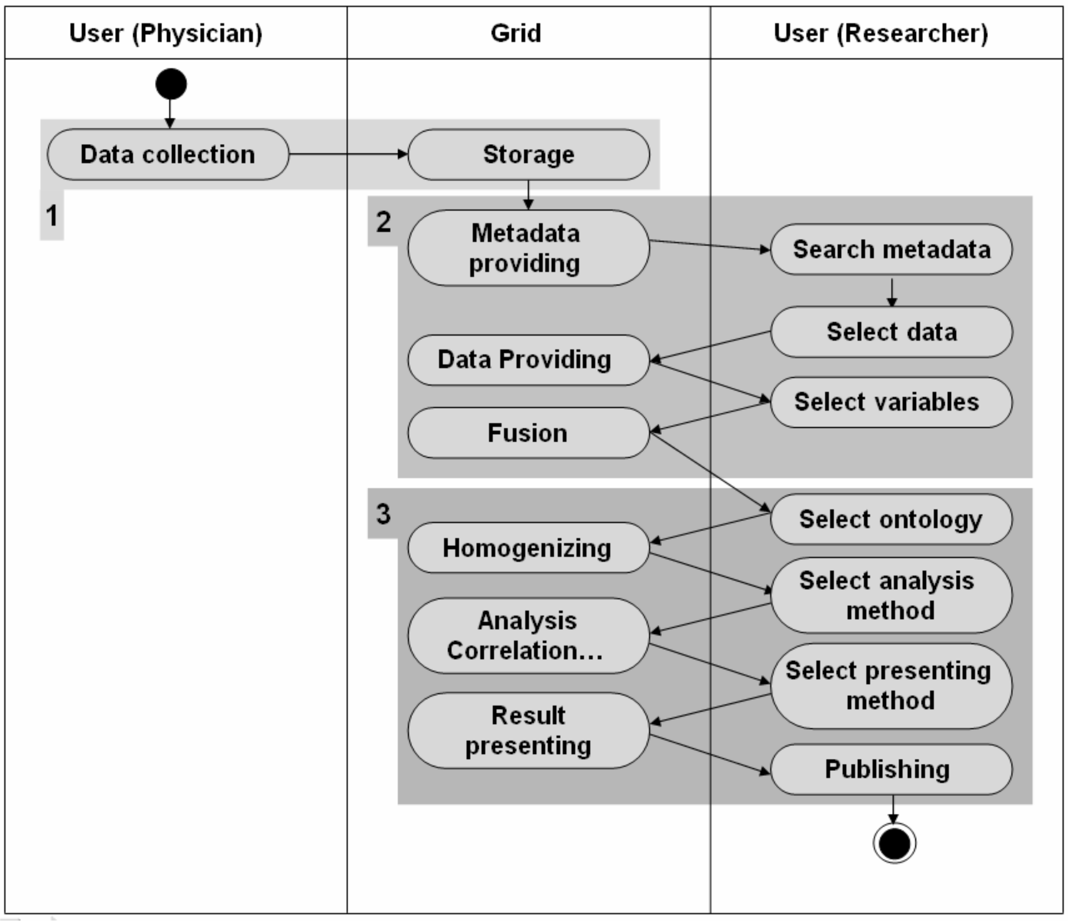


Figure 22: Activity diagram of the service flow in MediGRID as an example for a HealthGrid: 1- Upload on the grid 2- Retrieval: the user (researcher) can retrieve and select the data he needs for his work or research - the researcher prepares the data for processing, anyhow the data itself is not changed yet, 3- Processing: here the researcher will use algorithms and processing power available on the grid to process and analyze the data intending to receive the needed results [65].

Table 5: The five layers in Grid Security Infrastructure (GSI) as presented in GT4 [32]

Authorization	Grid-Mapfile/ SAML (Security Assertion Markup Language)
Delegation	X.509 Proxy Certificates
Authentication	X.509 ID Certificates
Message	WS-Security/ WS-SecureConversation
Message Format	SOAP (Simple Object Access Protocol)

The Grid Security Infrastructure (GSI) builds only the core for security in the GT4 middleware. GT4 uses the GSI to provide data management tools [168] for:

- data movement including GridFTP and Reliable File Transfer (RFT),
- data replication including Replica Location Services (RLS), and
- higher level data services – Data Replication Services (DRS).

These tools comply with the confidentiality of communication and the data integrity requirement. In the service chain (Figure 22), this fulfills the data security requirements for the first step – the upload service.

The second step in a data processing workflow on a grid, the data retrieval, requires more comprehensive and advanced data management. Using GT4 and some standard plugins, we can achieve the requirements to some good degree. With tools like Storage Resource Broker (SRB) – a data grid management system – [218, 219] and Data Access and Integration Services (OGSA-DAI) [53, 220, 221], the data availability requirement could be fulfilled. The access control in these plugins still, anyhow, is not advanced enough to comply with the requirements (e.g. no explicit Role Based Access Control possibilities).

Some biomedical applications, like DNA sequencing, do need special data protection enforcement on the software implementation level, because lower layers (middleware, operating system) do not support such possibilities. For example, sequencing applications with real human DNA data need to apply “binning” techniques before processing of the data [16]. In such application, not only processing but also accessing and retrieving data should be enforced and redesigned to match the particularity of the application as well as the data.

At the data processing level – the third step (Figure 22) –, the implemented and later deployed applications in the grid should take care of the requirements like confidentiality when handling the personal data. The Web Services Resource Framework (WSRF)[21] [169, 222], besides Grid Resource Allocation Management (GRAM) [223] and Monitoring and Discovery System (MDS) [224], build a suitable

[21] WRSF is a web services convention developed mainly for grid computing in order to add the ability to *manage* data and not only to *access and manipulate* data like in standard web services

execution and information management framework within GT4, which provides the possibility to implement and run confidentiality-aware applications.

Security levels regarding the used data in a healthgrid

The module e-science in the MediGRID project carried out an analysis to identify the classes of the processed data regarding the security requirements. The planned application and the target end users of these data were also criteria of this analysis (see Table 6) [225].

Applications involving processing of any human data have to meet the regulatory requirements, encompassing data protection and data security. For example, the principles of confidentiality and privacy have to be respected at all stages in a grid service workflow. In classical medical applications within hospitals this still takes place under the umbrella of the physician-patient confidentiality. Research computing requires more efforts concerning the protection of this confidentiality. This is a challenge in grid computing environments, as every available grid node has to be assessed regarding its trustworthiness by using some kind of trust metrics [65, 225]. Such applications were identified to have high security demands.

Nevertheless, we do have also medical applications of low or no extra data protection or data security requirements[22], e.g. gene sequence prediction of animal data [37, 226]. Such applications include non-human data or anonymized data with no re-identification risk. For such cases, the security issues are mainly determined by the common demands of the grid environment and resource providers, i.e. no extra security and data protection requirements needed. Such applications were identified to have low security demands.

Between the two extremes – low and high security – there is a class of applications that demand extra data protection measures. These data protection measures do not necessarily need to be implemented as grid services. Anonymized datasets which comprise a risk of re-identification as well as pseudonymized human datasets

[22] This is true at least in MediGRID. The reason is that MediGRID was planned from the beginning to be a computing infrastructure for the community, thus we deal with various applications with different requirements (horizontal grid solution). Other life sciences grids tend to handle one application (vertical grid solution).

can indeed be used (e.g. for research purposes) after obtaining a suitable consent from the data owner (mainly patients). With special measures, which are carried out before uploading the datasets to a grid environment, these data can be used. Such applications were identified to have medium security level.

Table 6: Security levels and application classes in healthgrid environments

Processed data	Security Level	Application Classes	User
Non-human data	low	- basic research - knowledge bases - demo versions	- researcher - all - all
Anonymized human data, no risk of re-identification	low	- basic research - clinical research - demo versions	- researcher - researcher/physician - all
Anonymized human data with risk of re-identification	medium	- basic research - clinical research	- researcher - researcher/physician
Pseudonymized human data	medium or high	- clinical research - clinical application	- researcher/physician - physician
Patient data	high	- clinical application - telemedicine	- physician - physician/patient

Current data security and data protection solutions in healthgrids

Most healthgrid projects yet follow the common grid middleware by trying to enforce data security with less work on data protection. The efforts regarding data protection in grids are mostly attempts to solve special cases rather than to find a common solution. The French MEDIGRID project implemented fine grained authorization with respect to the relationship between the user and the organization [227-231]. According to the developer, the security module in the French MEDIGRID – sygn – was designed to be more efficient than the Community Authorization Service (CAS) developed by the Globus team [110] and than the Virtual Organization Membership Service (VOMS) [120]. The MammoGrid project [232, 233] handled security as a service 'on the grid' and build it on top of the GT4-GSI tools [234, 235]. The GEMSS project [236] considered data protection for the special cases of medical simulations and image processing on the grid [57, 237]. The Cancer Biomedical Informatics Grid (caBIG) defines comprehensive security solutions on top of GT4. The project includes user-, trust-, and VO-management, identity federation as well as access control policy management and possibilities for the integration between existing security domains and the grid security domain.

These later solutions are designed for the special legal framework in the U.S.A., i.e. the Health Insurance Portability and Accountability Act (HIPAA) [238].

Including the mentioned projects, healthgrids depend basically on one of the different security approaches mentioned in Table 2, which make these approaches limited. No approach offers a solution that fulfills the different data protection requirements.

Perspectives on a suitable solution

The mentioned approaches and technologies fulfill to a good degree the requirement for data security in the grid but not for data protection. Considering GT4 as a middleware and the common plugins[23] regarding data retrieval and processing as well as pre-uploading procedures like anonymization and pseudonymization[24], we can identify the following shortcomings:

- Releasing only necessary data: the need to release only portions of the whole is an already identified problem. Solutions and standards from HL7 address Role Based Access Control and xml-structured data to solve this problem. Anyhow, HL7 standards do not address the grid as the processing and storage environment [128, 131, 239].
- Accountability: we still lack suitable technical solutions to identify who did what, when and for what reason in the grid.
- Retain patient-physician confidentiality: when a physician shares patient medical data, she should be sure herself and be able to assure her patients in advance that no one will use these data to create a more complete medical record on them or use the data for any other purpose, which is not mentioned in the consent the patient signed.
- Accessibility: some medical data, like emergency data, should be always available for authorized access.

Several analyses were performed and resulted in the need for supplemental security elements for healthgrids [43, 62, 64, 65, 157, 240-244]. Security extensions have been discussed in MediGRID [156] , the biomedicine community grid project

[23] Like SRB and OGSA-DAI.

[24] Anonymization and pseudonymization are mainly to be accomplished before uploading and processing sensitive or patient data in the grid.

in the German national grid infrastructure D-Grid [153]. MediGRID calls these supplements the Enhanced Security package, which includes:

- Access rights and access control management and enforcement: The current access control on file level (e.g. Grid-Mapfiles) is not sufficient, as (HL7) structured medical documents [129, 245] provide different sections with different degrees of confidentiality. Biomedical applications need fine grained access control with respect to different access rights within structured medical documentations. Another dimension of the requirements of access to medical applications and data in the grid is time, i.e. the possibility to control access over time. Moreover, the dynamic change of access rights – like the revocation of a particular permission of a particular user – plays an important role in a dynamic environment like the grid. A detailed discussion about access control and authorization is included in sections 2.2 and 2.3 and a supplemental discussion follows in section 4.2 and in the conclusion chapter.

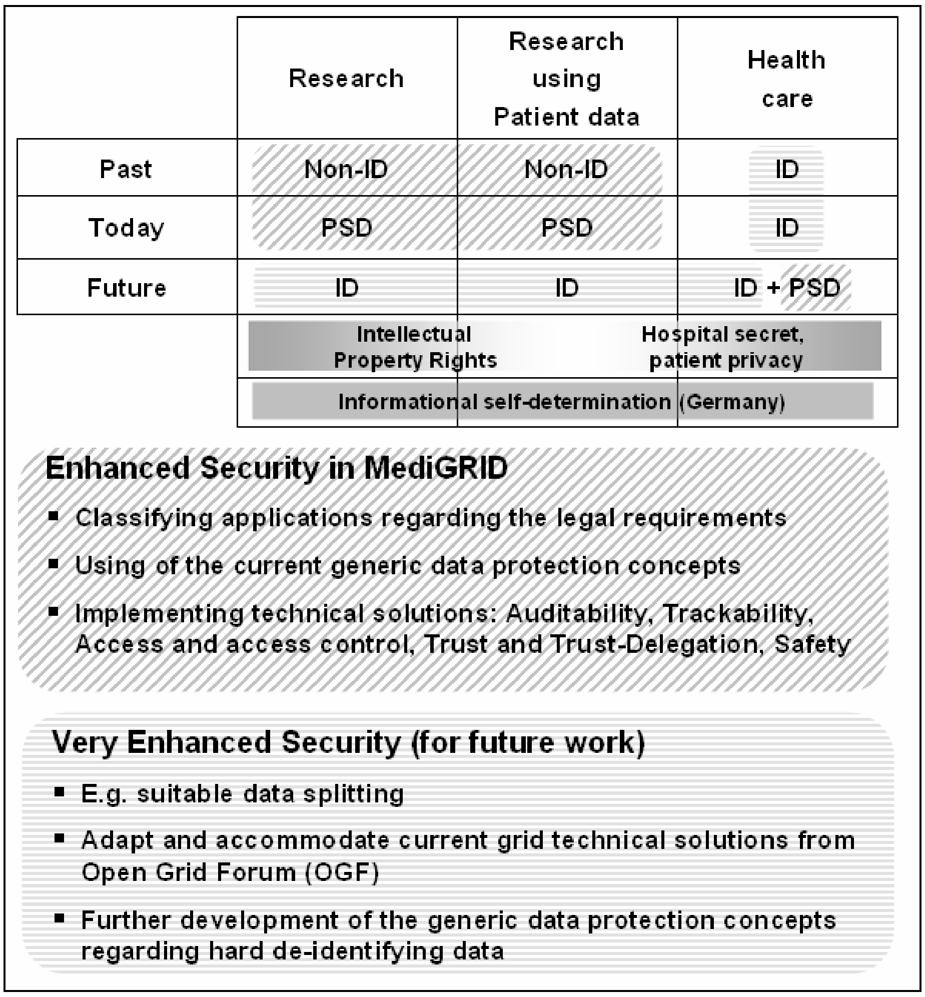
- Auditing possibilities are needed: Beyond the relevant data about the user, the application, the used datasets, and the used machine for each job, the validity of these information as well as valid time stamps are needed for an efficient audit. Further dimensions of auditing are data provenance and data annotation. While auditing is meant to be posterior to actions, a prior knowledge about where transfers, transactions, calculations, and storage of person related data take place is very important for healthgrids. In MediGRID this is called Trackability. Auditing and tracking possibilities should cover the requirement of accountability in the grid and retain the separation of identification data and medical data in order to preserve anonymity and/or pseudonomity. Many projects are working on the development of theoretical models as well as software for provenance and auditing on the grid [246-252]. Swift from GT4 developers is almost ready to be deployed and used in GT4. It records how every result is derived by the system [246, 247]. The U.K. e-Science provenance projects are more developed in theory [249-252]. Methods for preserving anonymity and pseudonomity for healthgrids databanks were also discussed [240].

- Trust relations and trust delegation as well as trust hierarchies from everyday life have to be set up electronically: Using the data of a minor or a person with dementia requires that an authorized person signs electronically on behalf of those persons (e-Consent). To have e-consenting possibility is very relevant for automatizing the data processing workflows, as the subject's (or patient's) e-consent is one of the bases for the authorization decision. Anyhow, the e-consenting workflows are described in some projects [253-255] and the implementation of such workflows for grid usage seems to be strait forward.

- Referred to it in MediGRID as "Safety", we need to develop and adopt suitable policies for the use and storage of data; a complementary safeguard principle when intending to use sensitive data and considering the availability in time (long term archiving) and place (replicas). The existing concepts and solutions for long term archiving and replicas still have to be implemented for grid computing environments. A most reasonable solution for replicas is to use the data management tools from GT4 [168], dCache [256] or SRB [218, 219]. A possible long term archiving solution could be to adapt the technologies developed by libraries for this purpose, i.e. by performing the integration on the level of the grid portal [158]. Anyhow, the different implementations still need to offer not only suitable storage and retrieval, but also suitable data deletion and disposal functionalities (again in time and place).

The elements of the Enhanced Security consider the current requirements of data protection and data security intending to make grid technologies more suitable for the biomedical community. In the future, supplemental security measures should also fulfill new legal requirements and new developments in the biomedical sector, e.g. Genome Wide Association Studies [257, 258] (see Figure 23: The re-identification of the object/patient becomes easier with additional datasets. The current legal framework allows the use of grid computing only with non-identifying data as well as pseudonymized data [62, 65]).

Different groups are working to overcome the mentioned technical shortcomings, mostly driven by the vision of having grid computing as a backbone for all kind of research and to think of it as the internet of the future. Nevertheless, from all security shortcomings in grid computing still authorization is the main problem, particularly for medical applications [43, 157].



	Research	Research using Patient data	Health care
Past	Non-ID	Non-ID	ID
Today	PSD	PSD	ID
Future	ID	ID	ID + PSD

Intellectual Property Rights — Hospital secret, patient privacy

Informational self-determination (Germany)

Enhanced Security in MediGRID

- Classifying applications regarding the legal requirements
- Using of the current generic data protection concepts
- Implementing technical solutions: Auditability, Trackability, Access and access control, Trust and Trust-Delegation, Safety

Very Enhanced Security (for future work)

- E.g. suitable data splitting
- Adapt and accommodate current grid technical solutions from Open Grid Forum (OGF)
- Further development of the generic data protection concepts regarding hard de-identifying data

Figure 23: The limits of using grid computing for medical applications. ID: identifying data, PSD: pseudonymized data [62, 65].

4.1.3. Perspectives on the grid computing security problem

The data security and data protection problem for medical applications in a grid computing environment is a many-sided and branching problem; it is neither only a technical problem nor is only a legal problem. With new technologies like grid computing, besides eventual new problems, old known problems get new shape. The general solution of these problems cannot be and will not be an effort of one party focusing or working towards specific implementation.

It is almost impossible to solve the problem of data security and data protection for the grid computing as a whole, especially in the biomedical sector. The intention in this work is to handle the most urgent problem in this regard theoretically, as it seems that much work is being invested in implementing the technology and building up silos for particular application (vertical solution).

The analysis shows that for healthgrids the authorization problem is the most urgent one. Not only because it is not solved yet, but also because the implementations and solutions yet still compare the grid authorization problem with the current established solutions in other environments, not taking into account the particularity of this new technology.

The following section will focus on this problem. The shortcomings of the current authorization systems for grid computing are described as well as steps towards a designing of a suitable authorization draft model, which could allow using grid computing for biomedical applications.

4.2. Shortcomings of current access control models regarding grid computing

4.2.1. Why current authorization models are not sufficient

In traditional access control, the enforcement of access has been primarily based on identities and attributes of already known users [21, 26]. In a dynamic distributed grid computing environment, digital information is to be used and stored at multiple sites. The information has to be protected regardless of the user location and the information source as well as storage locations. It is also required to control access by previously unknown users in order to have an open network-connected system. Both problems were solved partially by public key infrastructure and digital right management.

In spite of the adoption of RBAC through HL7 for access control to medical documents, healthcare continues to provide significant challenge for traditional access control, especially with the introducing of new technologies like Body Area Networks [259-261] or with building of new data banks for biomaterial data [262]. The HL7 vision of RBAC with the addition of structural and functional roles is meant to be applied within one domain, i.e. mainly inside the hospital. This approach will fail in a dynamic distributed environment, where the sharing of sensitive patient data goes beyond the hospital walls. In such scenarios, methods for maintaining privacy and property rights, for trust establishment and disclosure risk control are of great important.

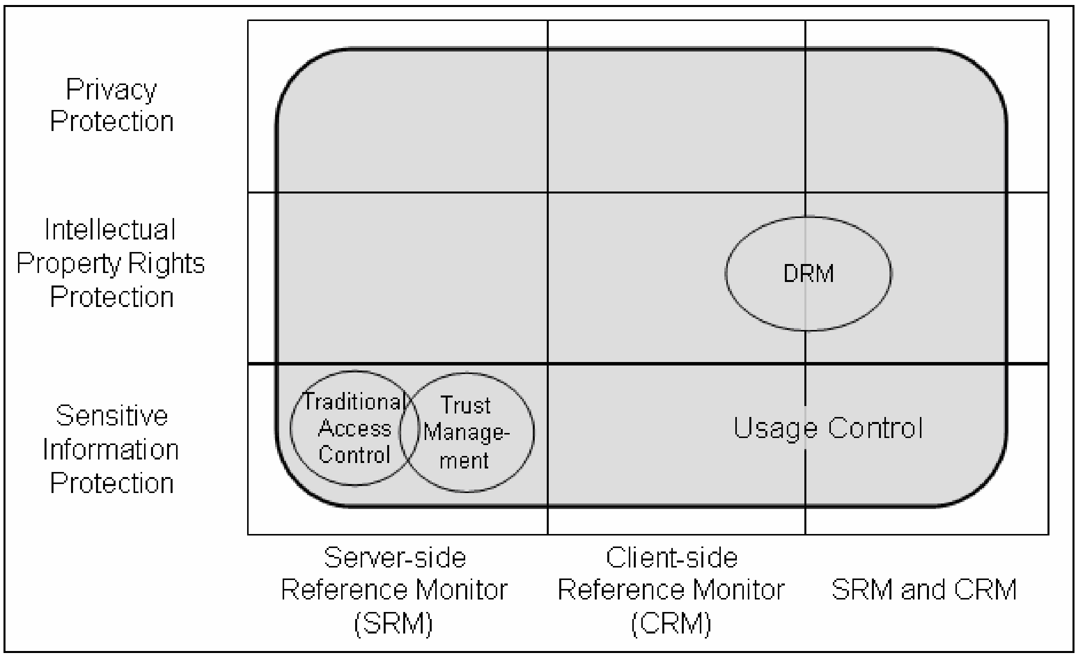
The different authorization systems developed or modified for grid computing are mostly driven by the traditional access control methods. These can indeed solve the access control problem of a particular application, i.e. vertical application. Anyhow, such a solution does not handle the grid authorization problem itself. Authorization polices that govern the interactions between VO members are currently expressed in terms of the identities of the individuals and resources [110]. They should also consider the entities roles in the VO as well as other complex states such as accounting and billing information, i.e. other attributes of the entities. Simple access control lists (ACLs) are not expressive enough for these complex authorization polices [32]. The different attempts to extend and/or redesign the existing access control systems exemplify indeed how current grid access control methods do not suffice [124, 127, 164, 229, 231, 263].

Granularity of the access is a practical problem with all current flavors of access control systems. This is due to the fact that operating systems work with files, which are usually the smallest objects, with which current access control mechanisms can deal. For example, we need application-level mechanisms to ensure that a researcher will see only the information inside a structured document (stored normally as one file), to which she has access permission [1]. Such application-level mechanisms are databank software. Nevertheless, new problems of fine-granularity appear when using databanks on the grid [240, 264].

Several researchers agree that classical access control needs extension, or even a redesign, to be adequate for modern applications. In the Chinese-Wall authorization model, the history of access is considered in the current and future authorization decisions [77]. In provisional authorization, authorization will be completed only when the subject carries out some action to make it effective [160, 265]. Task-based authorization treats right as a one- or n-time permission (consumable rights) [83]. In attribute-based access control, the authorization is based on user and object attributes. The existence of these attributes allows more flexible (fine grained) access control, rather than only the use of the subject identity. In the authorization model of XACML (an attribute-based access control standard from OASIS), the authorization policy administration is federated but the decision is centralized. In the Usage Control authorization model, the status of access to resources can change over time according to some policy, i.e. authorization is an ongoing process [25].

Referring to the latter approach, Park and Sandhu redefined in 2002 the authorization problem and introduced the Usage Control model (UCON) [23, 25, 146, 266]. One of the main advantages of the UCON model is that it handles the authorization as an ongoing process, where the status of access to subjects (resources) can change over time according to some policy. Moreover it adds extensions to model obligations, which should be fulfilled by the user and/or the system before, during, or after gaining access, and to model conditions, which should be present to gain the access. A distinguishing property of UCON beyond traditional access control models is the mutability of subject attributes and object attributes. Although mutability has its origins in traditional access control models, history-based access control policies can be expressed very good using UCON, where mutability is well defined [25, 144, 146].

Park and Sandhu argued that UCON is a conceptual framework, which provides a general-purpose, unified framework for protecting digital resources, and that "it encompasses traditional access control, trust management, and DRM and goes beyond in its scope" [25]. By having a detailed look at the UCON model, one can indeed comprehend the superiority of this model. The different examples presented by Park and Sandhu demonstrate how UCON can model different situations, which cannot be captured using traditional access control. According to the inventors, the UCON model does capture nearly all aspects of modern IT systems. The designers also provided examples on how to reduce UCON to all other known access control models. We can model RBAC, DAC or MAC systems using UCON. Figure 24 shows the coverage of the UCON model [25, 144, 145]. In this context, analyzing UCON means that all other known access control models are also considered.



Privacy Protection
Intellectual Property Rights Protection
Sensitive Information Protection
DRM
Traditional Access Control
Trust Manage-ment
Usage Control
Server-side Reference Monitor (SRM)
Client-side Reference Monitor (CRM)
SRM and CRM

Figure 24: The coverage of the Usage Control (UCON) authorization model. UCON encompasses traditional access control, trust management, and DRM [25].

Despite that UCON provides a more complete authorization system than other models, which makes it indeed more adequate for grid computing, it has shortcomings regarding its application in a grid computing environment. In the following subsection (4.2.2), critiques about and shortcomings of the UCON model regarding the grid computing authorization problem are discussed.

4.2.2. Shortcomings of the Usage Control model regarding grid computing

UCON is a branching transition system

One critique about the formal model of UCON is that it uses Linear Temporal Logic to describe a branching transition system. If we have a look again on the transition system of UCON in Figure 14, we can see that there are different states, which have more than one successor (see Figure 25). Hence, the use of linear logic is not adequate for this purpose. For the UCON transition system, one should use a Branching Temporal Logic. In the method chapter (chapter 3), we learned that the Alternating-time Temporal Logic is more adequate to describe such (open) transition systems.

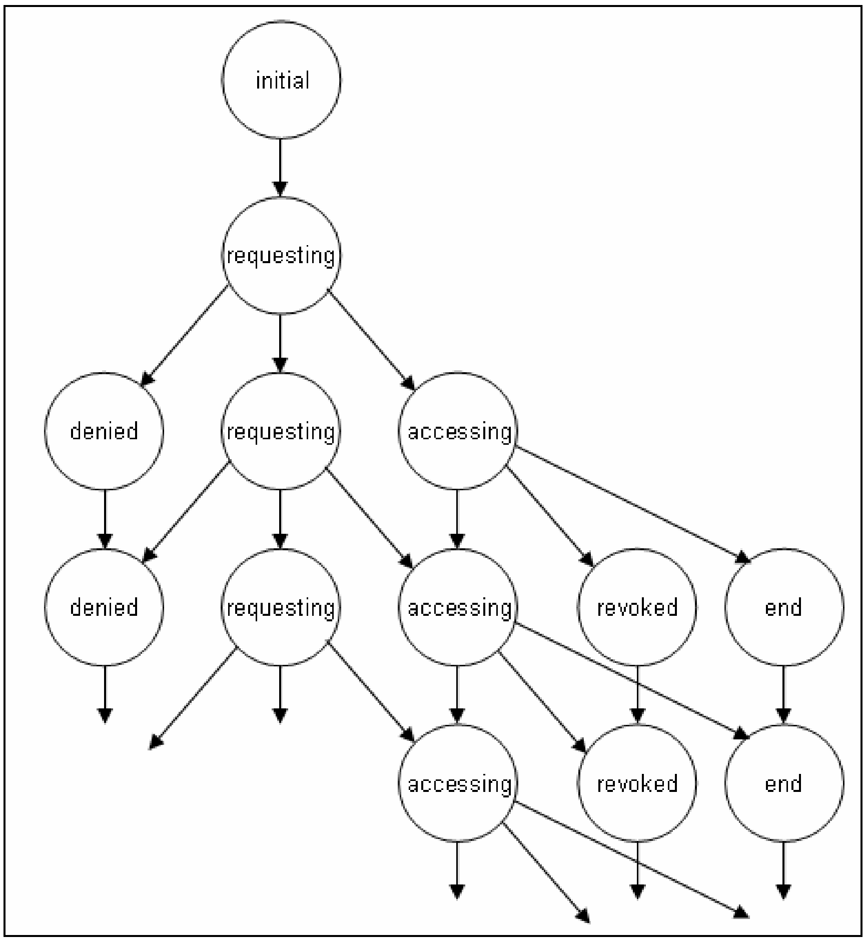


Figure 25: The Usage Control transition system is a branching and not a linear system.

UCON uses the Temporal Logic of Action

Another critique about UCON is the use of TLA. TLA a stuttering invariant temporal logic [267] and was originally developed to describe programs, not systems with components and architecture. For example, programs like *Pr1::x:=True; y:=False* and *Pr2::x:=True; Skip; y:=False* are not distinguishable by TLA specifications, but they are distinguishable by normal LTL [268]. Using TLA to describe an authorization system means that one process cannot consider changes done by other processes. This is in contrast to the properties mentioned by the inventors, in which an ongoing update of a usage can affect other usages [145]. The Alternating-Time Temporal Logic can capture this requirement.

Delegation is not included in UCON

In subsection 2.2.3 we discussed how essential privileges delegation in grid computing is and the efforts undertaken to extend current standards as well as to develop the proxy certificate standards in order to include this notion. All current grid authorization methods did (and actually should) implement the possibilities to deal with privileges delegation.

UCON was developed as a core model for authorization and “left delegation aspects for future development” considering it as “second order issues” [23, 25]. Privileges delegation could be a second order issue for many IT systems, especially when considering only human to human privileges delegation [163, 269-271]. For grid computing and because of the delegation role in it, the disregard of delegation is inadequate.

Privileges delegation in grid computing is the very first step in the authorization process. The delegation process should be a part of the authorization model from the beginning (here delegation is meant as delegation from human to a software instance).

UCON is designed for closed systems

In system modeling, it is important to use a formal model with a sufficient expressive power for that system. The UCON formal model is introduced using the Temporal Logic of Action (TLA) [144, 145]. TLA is extended from temporal logic by introducing boolean valued actions [148]. It is a linear temporal logic [148, 149, 272]. The classical temporal logics, whether linear (LTL) or branching (CTL) offer a natural model for the computations of closed systems, whose behavior is completely determined by the state of the systems. The compositional modeling and design of reactive systems requires each component to be viewed as an open system, where an open system is “a system that interacts with its environment and whose behavior depends on the state of the system as well as the behavior of the environment” [184, 273]. The main problem with approaches based on classical temporal logics is that such logics are not able to express the independences between the involved actions. Thus, the procedure synthesizes a global transition system and not a distributed concurrent one. Moreover, the constructed transition system might not be distributable (i.e. there is no distributed system exhibiting the same behavior) [274].

To model the authorization problem in a grid computing environment correctly, it should be considered as an open system. The different entities participating in the authorization decision work independently. Moreover, each resource in the grid should enforce the usage/access decision locally, and its behavior in doing this enforcement is not controllable. This problem is already known in the grid computing community as the “accounting and billing problem”: we have no control over the different resources in order to guarantee the trustworthy of the reported accounting and billing information.

In the UCON model, all actions are being carried out by the authorization system alone or by the user alone [144, 145]. In a grid environment, this is not applicable. We can identify this already in current grid authorization systems; although they use traditional access control models, many entities participate in the access decision (see Figure 7).

4.2.3. Recent proposals for grid authorization models based on the Usage Control model

Very recently, in the years 2006-2008, UCON has been discussed to be a possibly more suitable authorization model for grid computing. At the time of writing this work there were three articles addressing the issue of using UCON in its original definition – without handling its shortcomings- for grid computing [275-278].

Usage Control approach for Grid Services

In [276], Mei et al. introduce a Usage Control approach for Grid Services (UCGS). In this paper, the authors suggested to “extend” the grid infrastructure to become adequate for the UCON model. Their approach handles – theoretically – only the aspect of the subject’s (user’s) post obligations. For this purpose Mei et al. introduce the concepts of ″blacklist″, ″unilateral contract″ and ″arbitrator″ in their model. The authors consider that the grid environment – as a whole – is one system which has to ensure that users do their post obligations. This aspect UCGS is inherited from UCON. The UCON model considers the system as a closed one – a one player game in the game theory terminology, i.e. a game between one system and its environment. This makes UCGS – again, like UCON – limited to closed systems. UCGS introduces also four trust levels (TrustLevel): highest, high, low, and lowest. The rules to access a grid service will be determined according to the trust level between the subject and the object. With this, UCGS is reducing the authorization problem in grid environments to the traditional Discretionary Access Control problem.

Pervasive Computing Context Access Control based on the UCON 0odel

Pu et al. propose in [277] the use of the UCON model for pervasive computing and introduce a context access control model (CACM). Because one could consider pervasive computing as a special kind of grid computing (sometimes referred to as sensor grid), the work of Pu et al. should be considered here. The authors address UCON to be a better authorization model for pervasive computing; they include only three elements from UCON, namely pre-authentication, pre-condition and pre-obligation. The authors “emphasize on immutable attributes” [277] in their model. The ongoing and post authentication, conditions, and obligations are not included, which actually reduces the method to be a traditional access control system in the sense of granting static rights to the subject after she fulfills the obligations and under special defined conditions.

A usage-based authorization framework for collaborative computing systems

The most important proposal to use UCON for grid computing was made by Zahng et al. in [278, 279]. The authors provide a very comprehensive discussion of how to use the aspects addressed in UCON in a grid computing environment – according to the authors’ understanding. Nevertheless, and despite they mentioned the point briefly in their paper, they underestimate the importance of having multiple administrators in VOs and multiple resource providers; “Administrators in the VO and resource providers are also subjects that can define or change security policies. In this paper we focus on the control of general resources where the subjects are resource consumers, while the administrative aspect of the system is not considered.” [278]. Not considering the role of these two parties makes UCON – in its current definition – indeed applicable for the system, reduces anyhow the system itself to a closed system (rather a cluster computing system and not a grid computing environment anymore).

4.3. Proposal for a suitable authorization for medical applications

We learned in the previous sections and chapters that traditional access control systems are inadequate for modern distributed computing. The UCON model is the most modern and mostly complete theoretical authorization model, which encompasses traditional access control, trust management, and DRM. Anyhow, UCON does not define a delegation model and it is designed for closed systems. In the terminology of the game theory: UCON is a game between the user and the system where each transition is decidable by the user alone or the system alone. But even this situation is not describable using the UCON formal model. We learned also in the primer (sections 2.1 and 2.2) that the distinguishing property of a grid computing environment is that VO management systems and resource providers are administrative parties in the authorization process and that their decision is essential. The roles of these two parties are represented in the conventional grid environment by mapping agents for users and resources.

Putting all together, we can define the requirements for a suitable authorization model for grid computing, which we may call Grid Usage Control (G-UCON) model, as the following (see Figure 26):

- G-UCON includes UCON (which means G-UCON can model UCON),
- G-UCON is designed for open systems,
- G-UCON models delegation, and
- G-UCON models a user mapping agent(s) and a resources mapping agent(s).

In the following, G-UCON is described in more details. While the aim here is to define a more suitable grid authorization model, it is meant also to stay abstract. A good starting point is to identify the interacting entities and what distinguishes the events/actions and states chain regarding the authorization process in grid computing environment.

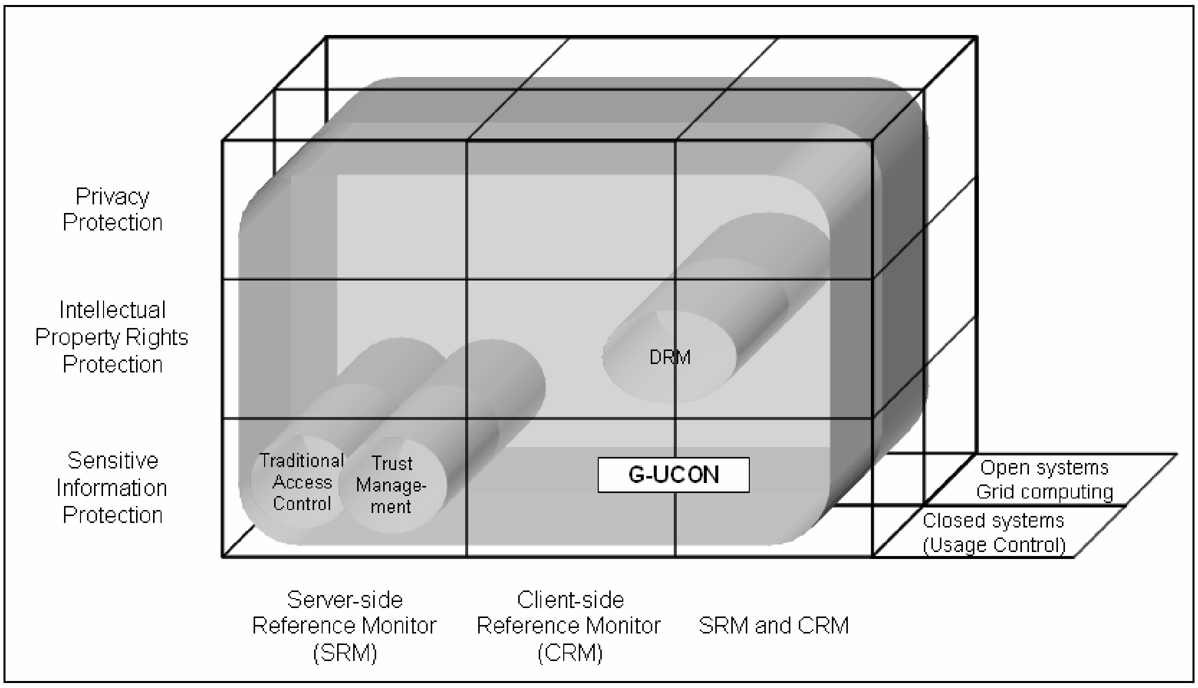

Privacy Protection
Intellectual Property Rights Protection
Sensitive Information Protection
Traditional Access Control
Trust Management
DRM
G-UCON
Open systems Grid computing
Closed systems (Usage Control)
Server-side Reference Monitor (SRM)
Client-side Reference Monitor (CRM)
SRM and CRM

Figure 26: The scope of G-UCON. G-UCON is designed for grid computing and open systems. It still can simulate closed systems.

4.3.1. The Grid Usage Control model: a more complete authorization model for grids

The interacting entities in G-UCON

The goal here is to capture an abstract description of the entities, which exist in a grid computing environment and interact during the authorization process. In G-UCON, beside the **User** there are three groups of systems (see Figure 27 and Figure 28), which participate in each authorization process:

- **UserSys**: a kind of software which works on the user infrastructure and manages the delegation of rights. ([25][280-282]).
- **BrokeringSys**: mainly a brokering system, which is responsible for mapping the abstract resources description in the user request to the available real resource after a successful authorizing at this level. It can also be a VO management system (the *CanUse* context [35]).
- **ResourceSys**: the real resources (nodes), which handle and execute the users' jobs and requests. The ResourceSys is responsible for performing the mapping of the grid user to a local user account, i.e. assigning the grid user job/request to

[25] MediGRID uses the MyProxy Upload Tool developed for the DataPortal project from the Council for the Central Laboratory of the Research Councils (CCLRC) within the UK e-Science program. User friendly-grids need such a system on the user side.

a local user on that machine. This mapping happens after a successful authorizing at the end resource level (the *CanLogin* context [35])

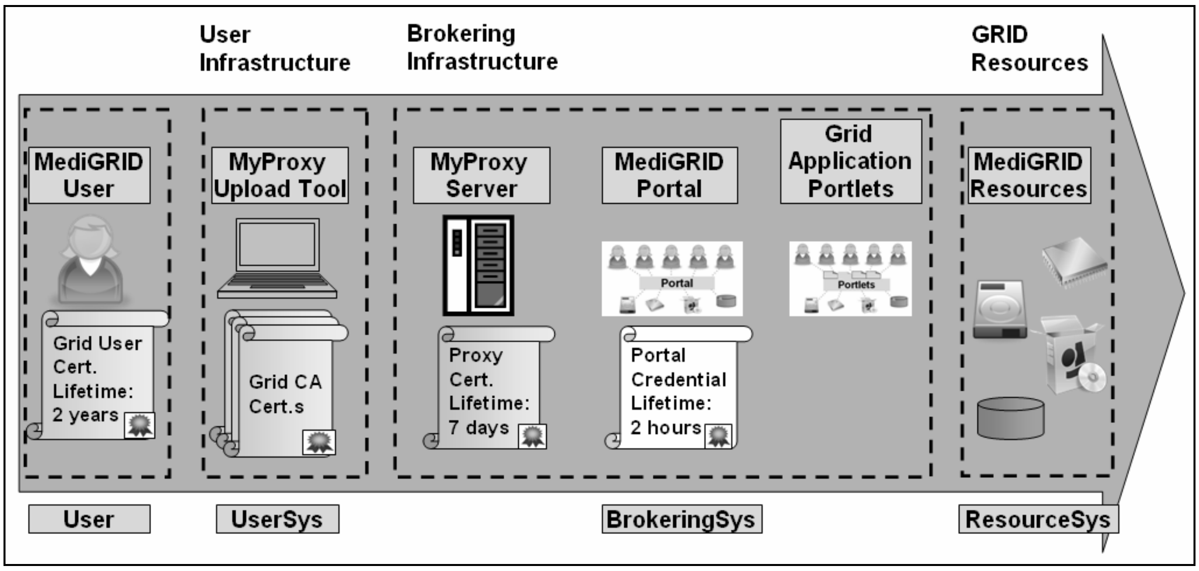


Figure 27: The participating entities in an authorization process in MediGRID as an example of a horizontal healthgrid environment. Each step needs to use additional trustworthy information (in a form of an x.509 signed certificate), which is the result of an interacting between different systems in order to establish a collaboration in order to authorize a user to access and use grid resources.

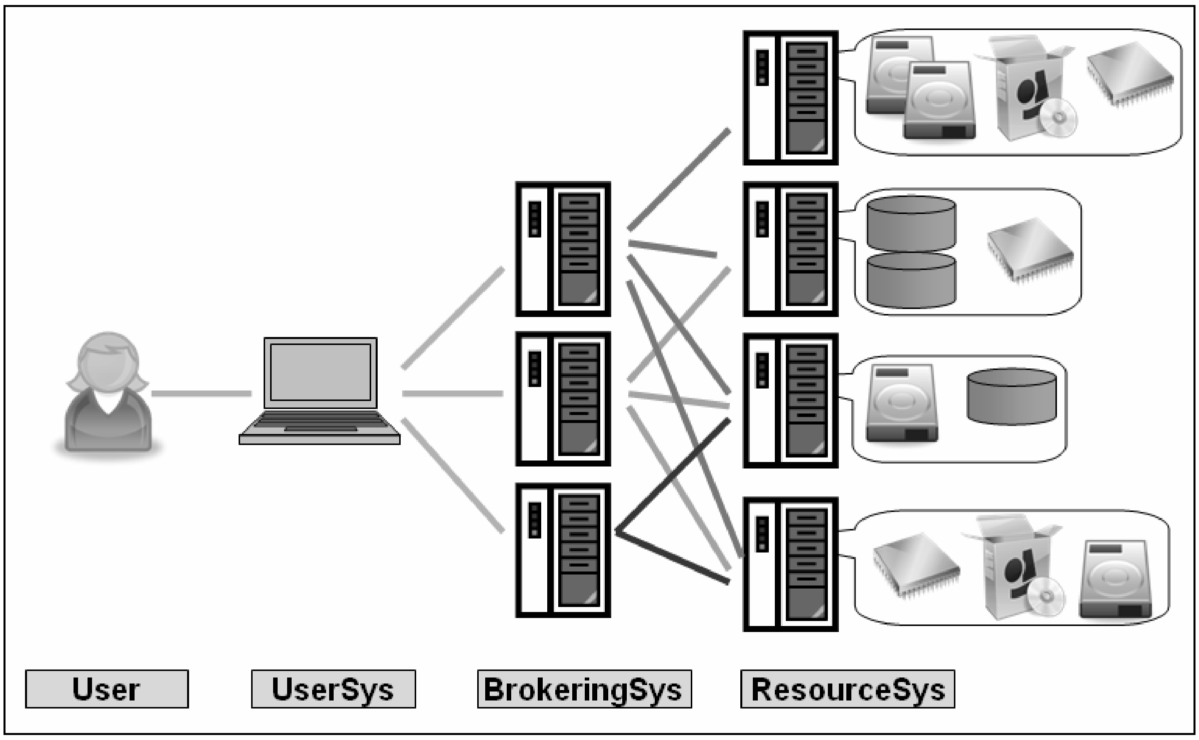


Figure 28: An abstract view on the participating entities in an authorization process in a grid computing environment. In a simple grid there will be at least four entities, which are independent in taking decisions.

G-UCON system states

Similar to UCON, each entity in G-UCON is specified by its attributes. Being in a specific state means that all attributes of an entity are assigned values (from their corresponding domains). A special attribute ***CurrentState*** $\in$ ***{Init, Delegating privileges, Failed, Requesting, Denied, Resource mapping, User mapping, Using, Revoked, End}*** specifies the usage state (see Figure 29).

In the following: ***subject1***: is normally the user, ***subject2***: is the process working on the user's behalf, ***rights***: is a vector of rights, ***object***: is the target object, ***process***: is the process to be performed on the target object ***object***. Processes include a description of the needed resources to run the process (applications consist in general of multiple processes that cooperate in some way), ***process.aresource***: is an abstract description of the needed resources to perform the process ***process***, ***resource***: is a description of real resources, ***localAccount***: is a local account on a real resource, and ***attribute***: is a vector of an entity/entities attributes (users and/or objects and/or system attributes…).

The semantics of the different states is as following:

- The ***init*** state is the starting state.
- The ***Delegating privileges*** state means that the rights delegation ***(subject1, subject2, rights)*** – normally from subject1 to subject2 – has been generated and is waiting for the (system's or systems') decision about this delegation.
- ***Failed*** means the delegation process ended with a failure.
- Being in the ***Requesting*** state means that the delegation ***(subject1, subject2, rights)*** was successful, the usage request ***(subject2, process(object))*** has been generated, and the primary authorizing (or brokering) system should search for a suitable real resource, which matches with the abstract resource description in the request.
- The ***Denied*** state means that the primary authorizing (or brokering) system has denied the usage request according to some policies.
- ***Resource mapping*** means the primary authorizing (or brokering) system found a suitable real resource that matches the abstract resource description in the request ***(subject2, process(object))***, and now the target node is handling the

mapping of the process to the real resources that match the abstract resources description ***(subject2, process.aresoure, resource)***.

- Being in the ***User mapping*** state means that the end target node mapped the request to a real resources (like a specific CPU), and the request to map the user to a local account is generated ***(subject2,localAccount,process(object))***.
- ***Using*** means the mapping was successful and the process ***process(object)*** is running now as a local process.
- A usage will be ***revoked*** when it is canceled by one of the participating systems (**ResourceSys** or **BrokeringSys**) during the ongoing usage.
- An ***End*** state will be reached when the subject (the **user** or the process working on her behalf) finishes her/its usage.

G-UCON actions (events)

The actions performed in G-UCON are of tow categories: actions involving one actor and actions involving multiple actors (see Figure 30).

Action involving one actor:

- ***TransferPrivileges(subject1,subject2,rights)***: transfer the needed privileges (***rights***) to the entity (***subject2***), which will work on behalf of the user (***subject1***). In the grid, this is a human-to-machine privileges delegation and is about creating the proxy and the proxy certificate. This action is performed by the user (***subject1***). It could also be a machine-to-machine delegation (precisely a process-to-process in the context of grid computing). The latter happens when a process "***call_new_process***" is called. The latter functionality is not described at this stage and is left for future work.

- ***DenyDelegation(subject1,subject2,rights)***: reject the delegation process (for example because of a wrong passphrase or undelegable privileges). Performed by the user's primary system (**UserSys**).

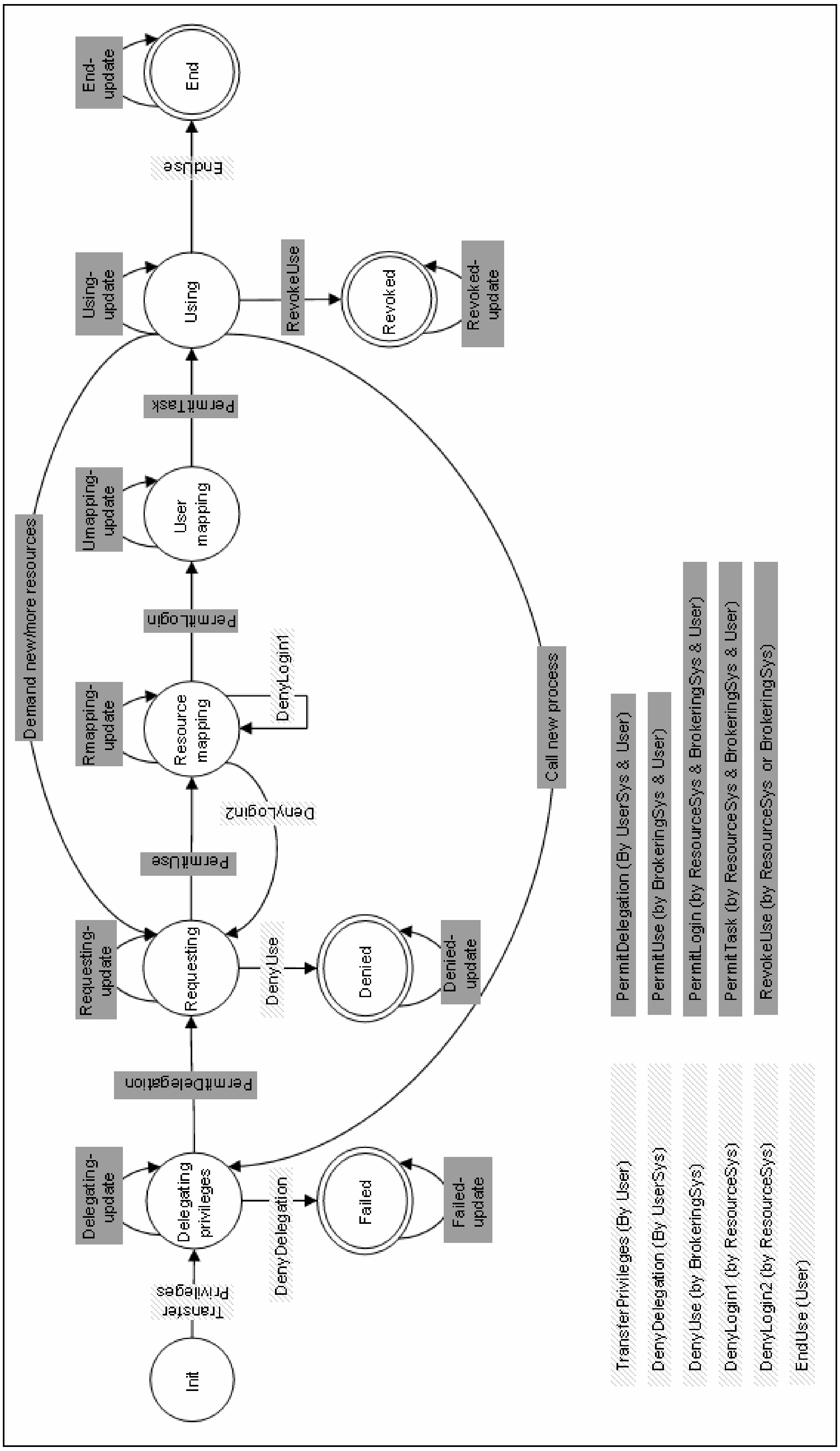

Init
Transfer Privileges
Delegating privileges
Delegating-update
DenyDelegation
Failed
Failed-update
PermitDelegation
Requesting
Requesting-update
DenyUse
Denied
Denied-update
PermitUse
DenyLogin2
Resource mapping
Rmapping-update
DenyLogin1
PermitLogin
User mapping
Umapping-update
PermitTask
Using
Using-update
RevokeUse
Revoked
Revoked-update
EndUse
End
End-update
Demand new/more resources
Call new process
TransferPrivileges (By User)
DenyDelegation (By UserSys)
DenyUse (by BrokeringSys)
DenyLogin1 (by ResourceSys)
DenyLogin2 (by ResourceSys)
EndUse (User)
PermitDelegation (By UserSys & User)
PermitUse (by BrokeringSys & User)
PermitLogin (by ResourceSys & BrokeringSys & User)
PermitTask (by ResourceSys & BrokeringSys & User)
RevokeUse (by ResourceSys or BrokeringSys)

Figure 29: State transitions and actions (events) in G-UCON. "Demand new/more resources" and "Call new process" are not described at this stage and are left for future work.

- ***DenyUse(subject2,process(object))***: reject the request of ***subject2*** to use/access ***object*** and to perform the operation ***process***. At this level, the description of the needed resources is still abstract and no explicit resource is mentioned. Performed by a primary authorizing system (like VOMS) or by a grid Broker (**BrokeringSys**). A reason to reject the request is for example that the user does not belong to a suitable VO to perform the submitted job, the defined job in the request is erroneous, or the abstract resource description is incorrect.

- ***DenyLogin1(subject2,process.aresource,resource)***: reject the mapping of the abstract resource to a real resource with the possibility to grant this request in the future (for example because of a maximum load when the request happens). Performed by the target real resource (**ResourceSys**).

- ***DenyLogin2(subject2,process.aresource,resource)***: reject the mapping of the abstract resource to a real resource without the possibility to grant this request in the future (for example, because of some enforcing policy at the target resource side, which does not allow this specific user (or VO) to use the recourses). Performed by the target real resource (**ResourceSys**).

- ***EndUse(subject2,localAccount,process(object))***: ends the usage *(subject2,localAccount,process(object))*. Performed by the **user**.

Action involving multiple actors:

- ***PermitDelegation(subject1,subject2,rights)***: allow the delegation of the ***rights*** from ***subject1*** to ***subject2***. Here ***subject2*** is rather a proxy process. Performed by the **UserSys** and the **user**.

- ***PermitUse(subject2,process(object))***: grant the request of ***subject2*** to use/access ***object*** and perform the operation ***process***. The description of the needed resources is still abstract at this level and is included within ***process***. Performed by a first authorizing system (like VOMS) or by a grid broker (**BrokeringSys**) as well as the ***User*** herself.

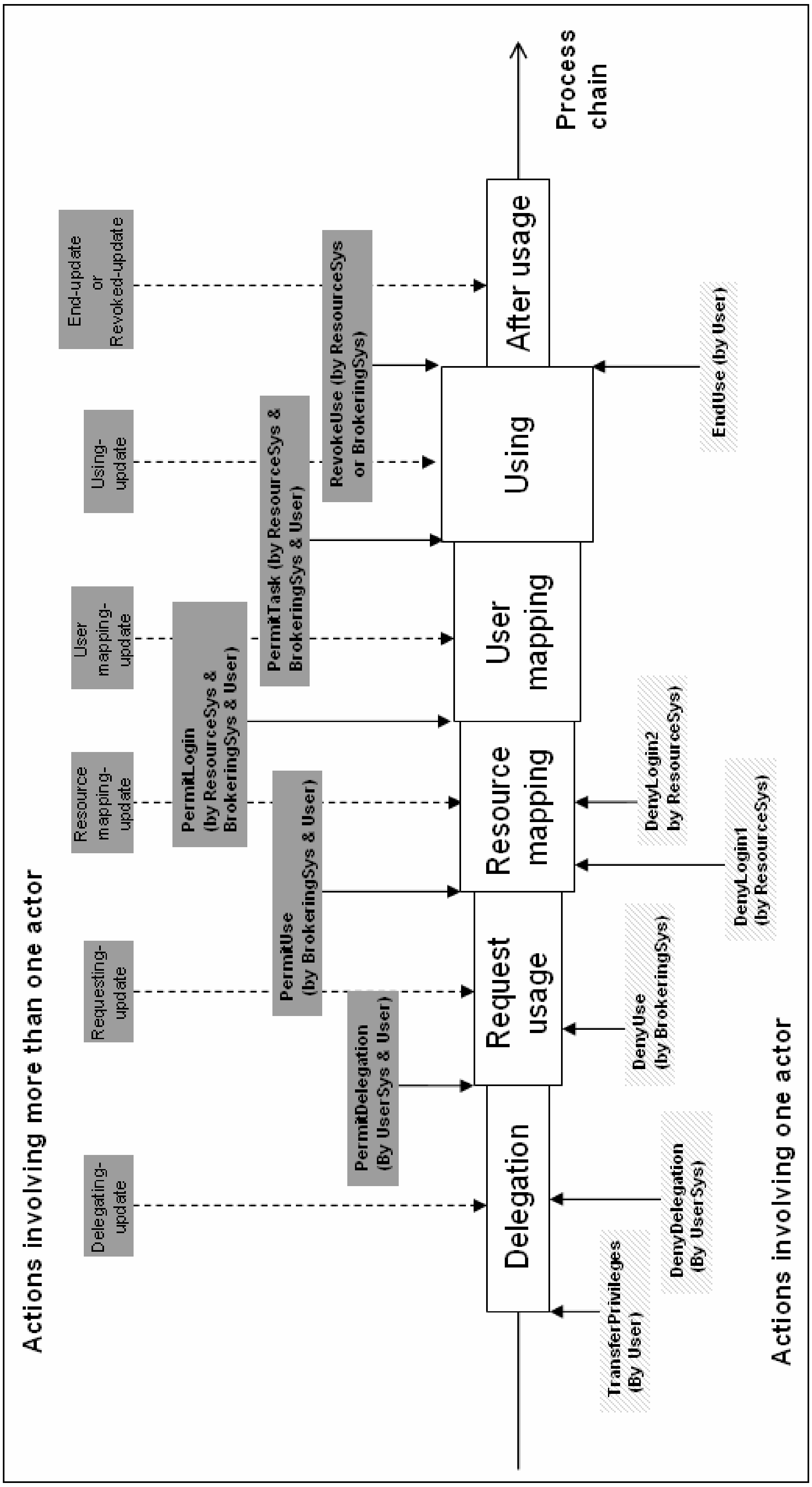
- ***PermitLogin(subject2,process.aresource,resource)***: allow the mapping of the abstract resource to a real resource. Performed by the target real resource mainly from grid broker (**ResourceSys**), but **BrokeringSys** and the **User** still can steer the process.

- ***PermitTask(subject2,localAccount,process(object))***: allow the mapping of the process working on behalf of the user to a local user account on the real resource. Performed by the target real resource **ResourceSys** with the possibility to be controlled by **BrokeringSys** and the user.

- ***RevokeUse(subject2,localAccount,process(object))***: revokes an ongoing usage. Performed by the first authorizing system (**BrokeringSys**) or by the target resource (**ResourceSys**).

- ***Update(attribute)***: updates subject's, object's, and/or system's attribute at the state when it is performed. Performed by one system or more (**ResourceSys**, **BrokeringSys**, or **UserSys**). There are nine different update possibilities in G-UCON, which are:
    - ***Delegating-update(attribute)***: during delegating privileges
    - ***Failed-update(attribute)***: follows an erroneous delegation process, i.e. during Failed state
    - ***Requesting-update(attribute)***: possibly during the requesting phase
    - ***Denied-update(attribute)***: eventually after an error in the requesting phase while the system is in the Denied phase
    - ***Rmapping-update(attribute)***: during the resource mapping state
    - ***Umapping-update(attribute)***: during the user mapping phase
    - ***Using-update(attribute)***: this is similar to the UCON onupdate. This is being carried out during the using phase
    - ***Revoked-update(attribute)***: this will be called if update is needed after the usage has been revoked and the system entered the Revoked state
    - ***End-update(attribute)***: after finishing the usage successfully, this action could be triggered to perform eventually needed post usage updates.

  The system, which performs an update, may require an input from one or more another systems to perform the update. That is why updates are actions involving multiple entities.



Actions involving more than one actor
Delegating-update
Requesting-update
Resource mapping-update
User mapping-update
Using-update
End-update or Revoked-update
PermitDelegation (By UserSys & User)
PermitUse (by BrokeringSys & User)
PermitLogin (by ResourceSys & BrokeringSys & User)
PermitTask (by ResourceSys & BrokeringSys & User)
RevokeUse (by ResourceSys or BrokeringSys)
Delegation
Request usage
Resource mapping
User mapping
Using
After usage
Process chain
TransferPrivileges (By User)
DenyDelegation (By UserSys)
DenyUse (by BrokeringSys)
DenyLogin1 (by ResourceSys)
DenyLogin2 by ResourceSys)
EndUse (by User)
Actions involving one actor

Figure 30: Continuity of the authorizing as well as the usage process and the different actions/events in G-UCON.

Designing G-UCON for open systems

In order to make G-UCON an authorization system for a grid computing environment, i.e. open system, G-UCON uses Alternating-Time Temporal Logic (ATL) in its definition. Using ATL, we can extend all UCON formulas by introducing the ATL path quantifier. So for example, the Grid Usage Control policy for the model UCON preA0 policy [23, 144] would be:

$$permitaccess(s,o,r) \rightarrow \langle\langle A \rangle\rangle \blacklozenge (tryaccess(s,o,r) \wedge (p_1 \wedge \ldots \wedge p_i)),$$

where $p_1, \ldots, p_i$ are predicates built from subject and/or object attributes, which are preauthorization predicates, and *A* is the set of entities (players) that can enforce the usage decision. In the UCON context, these players are the subject (s) who will fire the *tryaccess*, and the system that updates the subject and object attributes. In analogy, we can redefine the UCON model to be an authorization model for open systems and to be a building block of the G-UCON model.

G-UCON specifications

In the following a description of the core G-UCON specifications is given, but without the updates functionalities. This is meant to provide a general understanding of the G-UCON model and of how to define the different aspect of the model. It is not to be considered as the complete formal definition. Nevertheless, even though these specifications are still informal, they represent the core of the G-UCON model. Specifically, these specifications show how to model the needed delegation, to model the user mapping and resource mapping agents, to implement the UCON specifications, and how to keep the model applicable for open systems, precisely for grid computing systems. The goal here is to show the needed potential and different useful possibilities of modeling the grid authorizing aspects using G-UCON.

The given specification does not consider update possibilities. These possibilities are to be defined in analogy to UCON. Anyhow, instead of having three kinds of update (pre, on, post), we have nine different update possibilities in G-UCON: each corresponding to a different state (see Figure 29 and Figure 30). This allows us to consider a specific update process; for instance, to build a policy such that: when ***PermitUse*** happens, there should be update on the user attributes only if we reach the **Revoked** state (i.e. ***Revoked-update***). Such a policy is not possible to be implemented if we use only three kinds of updates.

The G-UCON model is a concurrent game structure with five players – in its simpler form, i.e. $k=5$ (actually we have five groups of players). We have the **User (or subject1),** the **Process (or subject2),** the **UserSys,** the **BrokeringSys,** and the **ResourceSys**. G-UCON has ten states ***Q={Init, Delegating privileges, Failed, Requesting, Denied, Resource mapping, User mapping, Using, Revoked, End}***. G-UCON uses the ATL temporal operators. The basic temporal operators include “Always”: $\Box$, “Eventually”: $\Diamond$, “Next”: $\bigcirc$, “Until”: $\mathcal{U}$, and the past operator extension “Has always been”: $\blacksquare$, “Once”: $\blacklozenge$, “Previous” $\ominus$, “Since” $\mathcal{S}$, and all are prefixed with the ATL path quantifier $\langle\langle\, \rangle\rangle$. This allows G-UCON to capture the open system characteristics of a grid computing environment. In the following $p_1,\ldots,p_i$ are predicates built normally from the different entities’ as well as environment’s attributes in the G-UCON.

We consider the current time point (t=0) as defined by the time when the described action itself is being carried out; i.e. future temporal operators refer to the time interval that starts by finishing the current action. Similarly, past operators refer to the time interval that ends by starting the current action.

The system attribute ***CurrentState*** returns the current usage state (one of ***{Init, Delegating privileges, Failed, Requesting, Denied, Resource mapping, User mapping, Using, Revoked, End}***).

TranferPrivileges

The user alone can transfer her privileges whenever she wants. Whenever we are in the init state, the user alone can enforce the transition to move to the Delegating_privileges state. And all other entities cannot avoid this transition[26].

- $\langle\langle user\rangle\rangle\Diamond\, TransferPrivileges\big(user, subject2, rights\big)\rightarrow$
  $\langle\langle\ \rangle\rangle\blacklozenge\,\big((includeSubject(subject2, rights))\ \wedge\ (p_1\wedge\ldots\wedge p_i)\big)$

- $\langle\langle user\rangle\rangle\bigcirc\,\big(CurrentState = Delegating\_privileges\big)\rightarrow$
  $\langle\langle\ \rangle\rangle\ominus\,\big(CurrentState = init\ \wedge$
  $TransferPrivileges(user, subject2, rights)\big)$

- $[\![\Sigma / user]\!]\bigcirc\,\big(CurrentState = Delegating\_privileges\big)\rightarrow$
  $\langle\langle\ \rangle\rangle\blacksquare\,\big(CurrentState = init\big)$

where $includeSubject(s,r) : Subjects' = Subjects \cup \{s\}\ \wedge\ s.reights = \{r\}$

When the user wants to limit the delegated privileges to the current job, the first part of the policy could be changed to:

- $\langle\langle user\rangle\rangle\Diamond\, TransferPrivileges\big(user, subject2, rights\big)\rightarrow$
  $\langle\langle\rangle\rangle\blacklozenge\,\big((includeSubject(subject2, reights)\ \mathcal{U}\ (Denied \vee Revoked \vee End)\big)$
  $\wedge\ (p_1\wedge\ldots\wedge p_i)\big)$

reasonably, $p_1,\ldots,p_i$ are predicates related to the ***TransferPrivileges*** action.

[26] The ATL formula $\langle\langle A\rangle\rangle\psi$ means that the players in A can cooperate to make ψ true (they can "enforce" ψ), the dual formula $[\![A]\!]\psi$ means that the players in A cannot cooperate to make ψ false (they cannot "avoid" ψ).

PermitDelegation

If it is demanded that special privileges (*rights)* should be delegated from one entity (*subject1*) to another entity (*subject2*) for a special time period then the *User* and the *UserSys* can cooperate to permit this delegation (PermitDelegation). Through this cooperation the current state will be changed to requesting. Other entities cannot cooperate to avoid PermitDelegation.

- $\langle\langle User, UserSys\rangle\rangle\Diamond\, PermitDelegation\big(subject1, subject2, rights\big) \rightarrow$
  $\langle\langle\ \rangle\rangle\blacklozenge\, \big((includeSubject(subject2, rights)\ \mathcal{U}\ (Denied \vee Revoked \vee End)\big) \wedge$
  $(subject1.id = User.id) \wedge (subject2 \notin Subjects) \wedge$
  $(rights \subseteq User.rights) \wedge (p_1 \wedge ... \wedge p_i)\big)$

- $\langle\langle User, UserSys\rangle\rangle\bigcirc\, \big(CurrentState = Requesting\big) \rightarrow$
  $\langle\langle\ \rangle\rangle\ominus\, \big(CurrentState = Delegating\_privileges \wedge$
  $PermitDelegation(subject1, subject2, rights)\big)$

- $[\![\Sigma / \{User, UserSys\}]\!]\bigcirc\, \big(CurrentState = Requesting\big) \rightarrow$
  $\langle\langle\ \rangle\rangle\blacksquare\, \big(CurrentState = Delegating\_privileges\big)$

where $p_1, ..., p_i$ are predicates related here to the ***PermitDelegation*** action.

PermitUse

After PermitDelegation and if all other requirements are fulfilled, the *User* and the *BrokeringSys* can cooperate to permit the *subject* to use the *process* on the *object*. This will change the state to Resource_mapping. Other entities cannot cooperate to avoid PermitUse. PermitUse represents the first authorization step through a VO management system or through the brokering system.

- $\langle\langle User, BrokeringSys\rangle\rangle\Diamond\ PermitUse\big(subject, process(object)\big)\rightarrow$

$$\langle\langle\ \rangle\rangle\blacklozenge\Big(CurrentState = Requesting\ \wedge$$
$$\big(subject.id = User.id \vee PermitDelegation(User, subject, rights)\big)\ \wedge$$
$$\big(subject.rights \subseteq process(object).neededRights\big)\ \wedge\ (p_1 \wedge ... \wedge p_i)\Big)$$

- $\langle\langle User, BrokeringSys\rangle\rangle\bigcirc\big(CurrentState = Resource\_mapping\big)\rightarrow$

$$\langle\langle\ \rangle\rangle\ominus\Big(CurrentState = Requesting\ \wedge$$
$$PermitUse\big(subject, process(object)\big)\Big)$$

- $[\![\Sigma / \{User, BrokeringSys\}]\!]\bigcirc\ \big(CurrentState = Resource\_mapping\big)\rightarrow$

$$\langle\langle\ \rangle\rangle\blacksquare\ \big(CurrentState = Requesting\big)$$

where $p_1,..., p_i$ are predicates related here to the ***PermitUse*** action.

PermitLogin

After a successful *PermitUse*, the user is authorized to use an abstract resource (*process.aresource*), she still needs the permission to use a specific real *resource*. Therefore, the *User*, the *BrokeringSys* and the *ResourceSys* will cooperate to permit the login to a specific node in the grid, which owns the needed *resource*. This will change the state to *User_mapping*. Other entities cannot cooperate to avoid *PermitLogin*.

- $\langle\langle User, BrokeringSys, ResourceSys\rangle\rangle\Diamond$

$$PermitLogin\big(subject, process.aresource, resource\big)\rightarrow$$
$$\langle\langle\ \rangle\rangle\blacklozenge\Big(CurrentState = Resource\_mapping\ \wedge$$
$$PermitUse\big(subject, process(object)\big)\ \wedge$$
$$process.aresources = resource \wedge\ (p_1 \wedge ... \wedge p_i)\Big)$$

- $\langle\langle User, BrokeringSys, ResourceSys\rangle\rangle\bigcirc\big(CurrentState = User\_mapping\big)\rightarrow$

$$\langle\langle\ \rangle\rangle\ominus\ \Big(CurrentState = Resource\_mapping\ \wedge$$
$$PermitLogin(subject, process.aresource, resource)\Big)$$

- $[\![\Sigma / \{User, BrokeringSys, ResourceSys\}]\!]\bigcirc$

  $\big(CurrentState = User\_mapping\big) \rightarrow$

  $\langle\langle\ \rangle\rangle\blacksquare\big(CurrentState = Resource\_mapping\big)$

where $p_1,\ldots,p_i$ are predicates related here to the ***PermitLogin*** action.

PermitTask

If the job is successfully mapped to a specific resource through *PermitLogin*, the user has the permission to use this specific *resource*. But to run a specific process or to perform a specific task, she still needs to be mapped to a local user. For this purpose, the *User*, the *BrokeringSys* and the *ResourceSys* will cooperate again to give the needed permissions to run the job/task as a local process on this resource (*PermitTask*). With this step we move to our main goal and the state will be changed to *Using*. Other existing entities cannot cooperate to avoid *PermitTask*.

- $\langle\langle User, BrokeringSys, ResourceSys\rangle\rangle\Diamond$

  $PermitTask\big(subject, localAccount, process(object)\big) \rightarrow$

  $\langle\langle\rangle\rangle\blacklozenge\big(CurrentState = User\_mapping\ \wedge$

  $PermitLogin(subject, process.aresource, resource)\ \wedge$

  $(p_1 \wedge \ldots \wedge p_i)\big)$

- $\langle\langle User, BrokeringSys, ResourceSys\rangle\rangle\bigcirc\big(CurrentState = Using\big) \rightarrow$

  $\langle\langle\ \rangle\rangle\ominus\big(CurrentState = User\_mapping\ \wedge$

  $PermitTask\big(subject, localAccount, process(object)\big)\big)$

- $[\![\Sigma / \{User, BrokeringSys, ResourceSys\}]\!]\bigcirc\big(CurrentState = Using\big) \rightarrow$

  $\langle\langle\ \rangle\rangle\blacksquare\big(\ CurrentState = User\_mapping\big)$

where $p_1,\ldots,p_i$ are predicates related to the ***PermitTask*** action; e.g. pre-use predicates built from the ***subject, localAccount, object,*** and ***process*** attributes (for example p= data storage on localAccount occurs with 1048 bit encryption).

DenyDelegation
If one (or more) of the requirements to permit the delegation of the subject1's privileges to subject2 is not fulfilled, the *UserSys* alone can deny this delegation immediately (in the next step). This will move us to the state *Failed*. Here it is very important to comment that other entities cannot cooperate to avoid *DenyDelegation*.   • $\langle\langle UserSys\rangle\rangle\Diamond\, DenyDelegation\big(subject1, subject2, rights\big) \rightarrow$ $\quad\langle\langle\ \rangle\rangle\blacklozenge\, \big(CurrentState = Delegating\_privileges\ \wedge\ (p_1 \wedge \ldots \wedge p_i)\big)$ • $\langle\langle UserSys\rangle\rangle\bigcirc\big(CurrentState = Failed\big) \rightarrow$ $\quad\langle\langle\ \rangle\rangle\ominus\big(CurrentState = Delegating\_privileges\ \wedge$ $\qquad DenyDelegation(subject1, subject2, rights)\big)$ • $[\![\Sigma / \{UserSys\}]\!]\bigcirc\big(CurrentState = Failed\big) \rightarrow$ $\quad\langle\langle\ \rangle\rangle\blacksquare\,\big(CurrentState = Delegating\_privileges\big)$ where $p_1,\ldots,p_i$ are predicates related here to the ***DenyDelegation*** action.

DenyUse
If one (or more) of the requirements to permit the use of *process* on *object* through *subject2* is not fulfilled, the *BrokeringSys* alone can deny this use immediately (in the next step). This will move us to the state *Denied*. Other entities cannot cooperate to avoid *DenyUse*.   • $\langle\langle BrokeringSys\rangle\rangle\Diamond\, DenyUse\big(subject, process(object)\big) \rightarrow$ $\quad\langle\langle\ \rangle\rangle\blacklozenge\, (CurrentState = Requesting\ \wedge (p_1 \wedge \ldots \wedge p_i)\big)$ • $\langle\langle BrokeringSys\rangle\rangle\bigcirc\big(CurrentState = Denied)\big) \rightarrow$ $\quad\langle\langle\ \rangle\rangle\ominus\big(CurrentState = Requesting\ \wedge$ $\qquad DenyUse\big(subject, process(object)\big)\big)$

- $[\![\Sigma / \{BrokeringSys\}]\!] \bigcirc \big(CurrentState = Denied\big) \rightarrow$

  $\langle\langle\ \rangle\rangle \blacksquare \big(CurrentState = Requesting\big)$

where $p_1, ..., p_i$ are predicates related here to the ***DenyUse*** action.

DenyLogin

If one (or more) of the requirements to permit the mapping of the user's request (job) to a real resource is not fulfilled, the target system (*ResourceSys*) alone can deny this mapping by either:

- temporarily deny (*DenyLogin1*) (for example because there is currently no free local account to map the job to). This will keep us in the *Resource_mapping*, or
- completely deny and move back to the *Requesting* state (*DenyLogin2*) giving the chance to find another suitable real resource.

DenyLogin1

- $\langle\langle ResourceSys \rangle\rangle \Diamond\ DenyLogin1\big(subject, process.aresource, resource\big) \rightarrow$

  $\langle\langle\ \rangle\rangle \blacklozenge\ \big(CurrentState = Resource\_mapping\ \wedge (p_1 \wedge ... \wedge p_i)\big)$

- $\langle\langle ResourceSys \rangle\rangle \bigcirc \big(CurrentState = Resource\_mapping\big) \rightarrow$

  $\langle\langle\ \rangle\rangle \ominus \big(CurrentState = Resource\_mapping\ \wedge$

  $DenyLogin1\big(subject, process.aresource, resource\big)\big)$

- $[\![\Sigma / ResourceSys]\!] \bigcirc \big(CurrentState = Resource\_mapping\big) \rightarrow$

  $\langle\langle\ \rangle\rangle \blacksquare \big(CurrentState = Resource\_mapping\big)$

where $p_1, ..., p_i$ are predicates related here to the ***DenyLogin1*** action.

DenyLogin2

- $\langle\langle ResourceSys \rangle\rangle \Diamond\ DenyLogin2\big(subject, process.aresource, resource\big) \rightarrow$

  $\langle\langle\ \rangle\rangle \blacklozenge\ \big(CurrentState = Resource\_mapping\ \wedge (p_1 \wedge ... \wedge p_i)\big)$

- $\langle\langle ResourceSys \rangle\rangle \bigcirc \left(CurrentState = Requesting\right) \rightarrow$

  $\langle\langle\ \rangle\rangle \ominus \left(CurrentState = Resource\_mapping \wedge\right.$

  $\left. DenyLogin2\left(subject, process.aresource, resource\right)\right)$

- $[\![\Sigma / ResourceSys]\!] \bigcirc \left(CurrentState = Requesting\right) \rightarrow$

  $\langle\langle\ \rangle\rangle \blacksquare \left(CurrentState = Resource\_mapping\right)$

where $p_1, \ldots, p_i$ are predicates related here to the ***DenyLogin2*** action.

RevokeUse

If one (or more) of the requirements to permit the use of *process* on *object* through *subject2* or the mapping to the local account is not fulfilled, the *BrokeringSys* and *ResourceSys* can revoke the use immediately (in the next step). This will move us to the state *Revoked*. Other entities cannot cooperate to avoid *RevokeUse*.

- $\langle\langle BrokeringSys, ResourceSys \rangle\rangle \lozenge$

  $RevokeUse\left(subject, localAccount, process\left(object\right)\right) \rightarrow$

  $\langle\langle\ \rangle\rangle \blacklozenge \left(CurrentState = Using \ \wedge \ (p_1 \wedge \ldots \wedge p_i)\right)$

- $\langle\langle BrokeringSys, ResourceSys \rangle\rangle \bigcirc \left(CurrentState = Revoked\right) \rightarrow$

  $\langle\langle\ \rangle\rangle \ominus \left(CurrentState = Using \ \wedge\right.$

  $\left. RevokeUse\left(subject, localAccount, process\left(object\right)\right)\right)$

- $[\![\Sigma / \{BrokeringSys, ResourceSys\}]\!] \bigcirc \left(CurrentState = Revoked\right) \rightarrow$

  $\langle\langle\ \rangle\rangle \blacksquare \left(CurrentState = Using\right)$

where $p_1, \ldots, p_i$ are predicates related here to the ***RevokeUse*** action.

EndUse

The user or the process working on her behalf can decide (alone) at any time to finish her/its ongoing use. This will move us to the *End* state. Other entities cannot cooperate to avoid *EndUse*.

- $\langle\langle User\rangle\rangle\Diamond\, EndUse\big(subject, localAccount, process(object)\big) \rightarrow$
  $\langle\langle\ \rangle\rangle\blacklozenge\,\big(CurrentState = Using \wedge$
  $PermitTask\big(subject, localAccount, process(object)\big) \wedge (p_1 \wedge \ldots \wedge p_i)\big)$

- $\langle\langle User\rangle\rangle\bigcirc\,\big(CurrentState = End\big) \rightarrow$
  $\langle\langle\ \rangle\rangle\ominus\,\big(CurrentState = Using \wedge$
  $EndUse\big(subject, localAccount, process(object)\big)\big)$

- $[\![\Sigma / User]\!]\bigcirc\,\big(CurrentState = End\big) \rightarrow \langle\langle\ \rangle\rangle\blacksquare\,\big(CurrentState = Using\big)$

where $p_1, \ldots, p_i$ are predicates related here to the ***EndUse*** action.

Writing policies for grid applications using G-UCON

The G-UCON specifications are meant to be flexible, i.e. modifications for the different use cases are mostly necessary. Most modifications are to be done in the different parameters of each action. The G-UCON actions and states are not modifiable without suitable modifications on the corresponding specifications. The flexible design of G-UCON makes extensions always possible. The written specifications are meant to match a principle grid computing environment consisting of the four groups of entities (see Figure 2, Figure 27, and Figure 28). A real grid computing environment may contains different other entities like separated Accounting and Billing entities, separated Privileges Storing and Retrieving entities, or Usage Monitoring entities. It is very likely to have such systems in an authorization system for a grid environment. G-UCON cannot model such entities without extensions to the used transition system as well as to the specifications. Anyhow, as a conceptional model, G-UCON should be kept as simple as possible on the one hand, on the other hand, it should handle the smallest possible grid computing environment.

How we can use G-UCON for writing authorization policies, which are not possible using the state of art models, will be discussed in the next chapter (5). A few examples of writing G-UCON policies are also presented.

5. Discussion, Conclusions, and Outlook

Gartner Inc., the known information technology research and advisory company, predicted in 2003 that “by 2008, SOA will be a prevailing software-engineering practice, ending the 40-year domination of monolithic software architecture” [283]. The grid computing activities were reformed in 2003 by introducing the third version of Globus Toolkit in order to adapt the Web Services (WS) and Service Oriented Architecture (SOA) technologies [32]. This adaptation was a strategic step for the developing of the grid technology to be an infrastructure for science [284]. In April 2006, the first grid web services standards – the Web Services Resource framework (WSRF) – were adopted and are still evolving [169, 222]. In the same year Gartner predicted also that “the lack of working governance mechanisms in midsize-to-large (greater than 50 services) post-pilot SOA projects will be the most common reason for project failure" [285]. In 2006 also, Manes, the vice president and research director in Burton group pointed out that “many organizations don’t start to think about governance until things are completely out of control” [286]. In this context, SOA governance is the process of defining and enforcing organizational policies and standards. Hence, it was clear that governance would be a problem in the different approaches to a SOA environment including grid computing. Nevertheless, limited efforts were invested to be ready to face this problem.

Authorization is comprised in IT governance. Weill and Ross define governance shortly as "specifying the decision rights and accountability framework to encourage desirable behavior in the use of IT” [287]. IT governance implies control and measurement in the computing environment. It also provides the framework, mechanisms and methodology for involving all interacting people, from those being supported to those who provide support [288]. Governance in grid computing lacks not only a well defined framework that facilitates the enforcement of needed policies; moreover it lacks a method for defining needed policies (like defining policies using computer logic in order to check their correctness). Simply put, in regard to security, grid computing lacks correct functional authorization mechanisms, but more important, it lacks a functional authorization model. The existence of different approaches to solve the authorization problem verifies this (see subsection 2.2.3). These approaches clearly reflect attempts to solve the problem on the mechanisms level without reworking the conception of the upper levels (see Figure 3). For medical applications the needed authorization polices are already complex [1, 25]. Deploying the applications in the dynamic open grid computing environment makes the authorization problem even more complex. Hence, a very important step in engineering a solution for the authorization in a grid environment is to develop a methodology for modeling and writing policies [142].

The conclusions of the performed analysis

The results presented in this work show that current grid authorizing systems do not fulfill the requirements of a common biomedical application deployed in a grid environment. These systems depend mainly on traditional access control models (see subsection 2.2.3) or on extensions which are valid only for a particular application [230, 237, 289]. Three reasons were identified for this deficiency (see subsection 4.2.1 for details):

- Current authorization models and systems do not consider open systems. The reason mainly is that such authorization models for open systems were not demanded yet. With the introduction of grid computing environments and multi-agent computing alike systems, new perspectives on authorization for open systems have become necessary.
- The currently used authorization models for grid computing do not consider time as a dimension in the authorization decision. As the last step in submitting a grid job is mapping to a local account in the target resource, we are still using the operating system's static rights for authorization (grid map-files). Even that we have mechanisms to include parameters about time (e.g. XACML); we do not have the model that enables us to write dynamic rights and policies in order to reason about these mechanisms.
- The adapted models for grid authorization systems cannot and do not consider grid delegation (see subsection 2.2.3). The delegation framework is well implemented on the mechanisms level and there is no possibility to establish a grid environment without these mechanisms. The lack of possibilities to model grid delegation makes the adapted models inefficient.

Current trends in authorization models

The recently published theoretical Usage Control model was proposed by Sandhu et al. to be the authorization system of the future IT systems – the web services era. UCON comprises the features of mostly all other authorization and access control models and is even more powerful. Bearing in mind the influence Sandhu had on the development of all yet known access control models, one can indeed trust his vision on the future of access control. UCON is an attribute based access control model, which considers the notion of dynamic rights (or mutable attributes) and adds to it two other important concepts: obligations and conditions for a specific authorization. Independent from UCON, the grid computing community discussed recently the notion of obligations for implementation within the grid computing middleware GT4 [172]. Having a closer look on the UCON model we can notice, on the one hand, the superiority of this model, but also, on the other hand, shortcomings in its current definition when applying it for grid computing. These shortcomings are chiefly attributed to the fact that UCON was designed from the beginning for web services and not for grid computing services. The extensions and attempts to use UCON for grid computing were discussed and deficiencies were provided in subsection 4.2.3.

The contribution of this thesis

In this work, the Grid Usage Control (G-UCON) model is presented as a solution outline for the authorization problem in a grid computing environment. This model was designed from the beginning to be capable of fulfilling the requirements of biomedical applications. Anyhow, G-UCON is not meant to be a complete solution for the grid authorization problem, rather a first step in defining a model, which should facilitate writing and proofing policies-in-large for a grid computing environment. G-UCON is defined by mean of the four required building blocks: (1) G-UCON includes UCON, which means that G-UCON inherits the flexibility of UCON (continuity of decisions, mutability, and certainly the other common authorization concepts) and is ready in nature for web services. In order to be adequate for the grid computing environment, (2) G-UCON is designed for open systems, (3) it models delegation, and (4) it models user mapping agent(s) and resources mapping agent(s).

The design of the G-UCON model puts main emphasis on the fact that different actors coexist in a grid computing environment and they work independently. Current grid authorization systems and the used models do not take this fact into account, which is the origin of inapplicability in grid computing environments. In the context of game theory, the authorization problem in grid computing is a concurrent game. The usage/access decision is not a decision of one party; it is the result of collaboration between different parties. Current authorization models do not consider collaboration possibilities (see Figure 31 and section 4.2). The G-UCON model as a main result of this work is defined to capture this requirement.

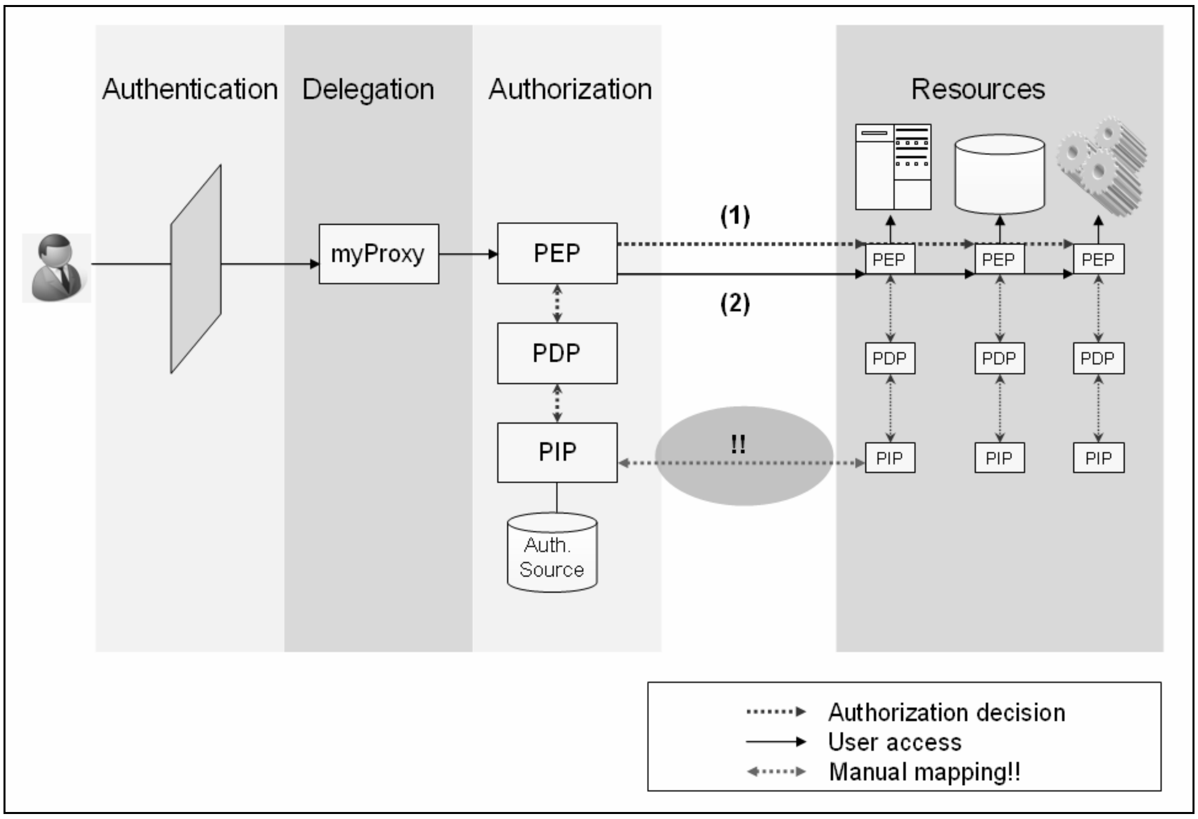


Figure 31: The lack of a mechanism to transfer the security policy information from the Policy Information Point (PIP) to the resources side and to enforce it there by connecting this information to the resources Policy Decision Point (PDP) and Policy Enforcing Point (PEP). Currently this is solved by manual mapping carried out by the resources administrators.

Before demonstrating how G-UCON can capture the latter mentioned requirements in section 5.2, the next section (5.1) provides a description of current possibilities to study the correctness of the designed model.

5.1. Validating G-UCON – possibilities and limitations

Automated reasoning, i.e. automatic model checking deals with automated methods to establish the satisfiability or validity (as well as unsatisfiability or invalidity) of sets of formulas. In order to perform a model checking (see subsection 2.3.1), we need a suitable transition system that can capture the properties of the modeled systems (like Kripke structure for closed systems), a suitable logic (like LTL for Kripke structure and linear temporal processes), and a suitable model checker for both (like SPIN [290]).

Alternating-Time Temporal Logic is the used logic

Alternating-Time Temporal Logic (ATL) is the common specification language for open systems. A closed system "is a system whose behavior is completely determined by the state of the system" whereas an open system "is a system that interacts with its environment and whose behavior depends on the state of the system as well as the behavior of the environment" [184]. ATL offers the possibility to handle the notion of alternating, i.e. to deal with the question whether the system can resolve its internal choices in a way that guarantees the satisfaction of a property, regardless of how the environment resolves the external choices. This is the main advantage of this logic. ATL allows the writing of formal specification of requirements that refer to collaborative relationships between entities [184].

Concurrent game structure is the used transition system

The formulas of ATL are to be interpreted using a concurrent game structure as a transition system. In ordinary transition systems, each transition corresponds to a possible step of the system. In a concurrent game structure, each transition corresponds to a possible move in the game between the participating players [184].

No available suitable model checker

The MOCHA tool is the automatic model checker for reactive modules and ATL over reactive modules [291]. Reactive modules allow formal specification of heterogeneous systems with synchronous, asynchronous, and real-time components. Reactive Modules support modular and hierarchical structuring and reasoning principles [292]. Anyhow, they are not identical with concurrent game structures [292]. No experiences or papers seem to be available that discuss similarities, differences, or possibilities to translate concurrent game structures into reactive modules.

ATL seems to capture all needed aspects to write the specification of a suitable grid authorization model, which is G-UCON in our case. Anyhow, recently and only via personal communication with the senior introducer of this logic – professor Alur from the university of Pennsylvania –, it became clear that the developers of this logic “did not consider the possibility for creation/deletion of agents” in their model checking tool – MOCHA [293]. Alur continued “I am not sure if anyone has studied this topic” [293]. This is not mentioned explicitly in any published literature, which was available during this research. Creation/deletion of agents is a very important aspect for grid computing because we have to consider grid delegation.

This is an example of how a new technology demands development in other disciplines in order to be adapted. When grid computing was introduced in the mid 1990s, the lead motivation was the increasing need for vast storage capacity. Soon after, the focus changed to supercomputing. Now, we are facing an increasing necessity to understand how this relatively new technology behaves, can be used or misused. The target goal is to be able to anticipate a future possible behavior of an entity in a grid computing environment in order to arrange accurate suitable reactions. Especially when authorization is concerned, a precision in anticipating behaviors as well as defining reactions is of a capital importance. This work captures these demands by the performed analysis and by the introduced outline for a possible solution, i.e. G-UCON.

The practice of presenting real world example for first evaluation

In nearly all access/usage control models including DAC, RBAC and UCON, a primarily validation was carried out by means of showing how the model fits to real world examples; especially examples from the different situations that motivated the design of the model. Implementations (as software) appeared years later after the development of the model and allowed a practical validation. For instance, the RBAC model was published in 1992, a first implementation within an operating system appeared in 2000 in Solaris 8 [90], next after implementation were in Windows Server 2003 [82] and Red Hat Enterprise Linux version 5 in 2007 [89].

Currently, it is difficult to build a functional authorization system for a real world grid environment using G-UCON. The validity of the G-UCON model cannot be performed likewise. A validation using available automatic model checking tools is currently not possible because these tools do not allow us to consider a grid delegation process. Therefore, we restrict ourselves in this work to informal validation by providing multiple examples (see subsection 5.2), which show how G-UCON can capture the design objectives; primarily, modeling authorization for grid computing.

The contribution of this work in light of these limitations

G-UCON delivers a concept of minimal requirements for a suitable authorization model for medical applications deployed in a grid computing environment. This concept is formulated in a quasi-mathematical way, which can currently only provide an understanding of how an authorization model for a grid computing environment should look like and how such a model should be implemented. This work derives a model on the basis of an analysis of the authorization problems in grid computing environments. Although the proposed model is described using temporal logic, in the absence of a suitable validation tool, the used mathematical formulas remain a quasi-mathematical description. Anyhow, in this context we still can describe future steps to move ahead in order to validate this model. These are included in the subsection 5.3.

5.2. The expression power of G-UCON

As no formal model checking is currently possible, the following use scenarios and reduction examples provide a basis to understand how to use G-UCON as well as what makes this approach novel and grid-aware.

5.2.1. Modeling other access control models using G-UCON

UCON was considered from the beginning as a building block of G-UCON. On the specification level, G-UCON is defined using ATL, which is an upper set of the temporal logic used to define UCON. This makes it possible to model UCON using G-UCON.

The reduction of G-UCON to UCON is possible:

- by reducing the multi-player game of G-UCON to a one-player game, i.e. to consider that ***(the G-UCON) User*** = ***(the G-UCON) UserSys*** = ***(the G-UCON) BrokeringSys*** = ***(the G-UCON) ResourceSys*** = ***(the UCON) System*** (see Figure 29), and
- by considering a corresponding policy to the targeted UCON policy, i.e. to consider a trivial policy for all non-UCON actions (like ***TransferPrivileges*** or ***PermitLogin***). This reduces the G-UCON transition system to the transition system of UCON, i.e. without actions and states for delegation and mapping agents.

We need to keep in mind that all G-UCON policies deal with states and actions. UCON uses TLA and cannot capture the states of the transition system in its policies, i.e. we cannot reason about states using UCON policies. This is because TLA was designed to capture only actions, i.e. TLA is a stuttering invariant temporal logic (see subsection 4.2.2). G-UCON polices use ATL, which means that a rewriting of a UCON policy to be a G-UCON policy may change the complexity of the problem. Studying the complexity of the model concept is beyond the scope of this work.

We consider a one-player G-UCON authorization game. The ***System*** controls all G-UCON actions; therefore it can enforce the transition system to jump over all states that are not included in the UCON model (see Table 7). This means that the player ***System*** has also to jump over the actions that are not included in UCON (see Table 8). The reduction of G-UCON to UCON is illustrated by means of the following two examples.

Table 7: The different UCON and G-UCON states and the correspondences between them

Corresponding UCON state	**Corresponding G-UCON state**
Initial	*Init*
Requesting	*Requesting*
Denied	*Denied*
Accessing	*Using*
Revoked	*Revoked*
End	*End*
-	*Delegating Privileges*
-	*Failed*
-	*Resource Mapping*
-	*User mapping*

Table 8: The UCON and the G-UCON actions and the correspondences between them

Corresponding UCON action	**Corresponding G-UCON action**
Tryaccess	*PermitDelegation*
Permitaccess	*PermitUse*
Denyaccess	*DenyUse*
Endaccess	*EndUse*
Revokeaccess	*RevokeUse*
Preupdate	*Requesting-update and/or Denied-update*
Onupdate	*Using-update*
Postupdate	*Revoked-update and/or End-update*
-	*PermitTask*
-	*TransferPrivileges*
-	*PermitLogin*
-	*DenyDelegation*
-	*Failed-update*
-	*Delegating-update*
-	*Rmapping-update*
-	*Umapping-update*

Example 1: Reducing G-UCON to UCON *preA*$_0$

To illustrate the reduction of G-UCON to UCON, we consider the UCON *preA*$_0$ policy as an example. This policy is written using TLA and can express most traditional access control models [23, 144]. In *preA*$_0$, "an authorization decision is determined by the system before the access happens, and there is no update for subject or object attributes" [144]. The usage control policy for *preA*$_0$ is: $permitaccess(s,o,r) \rightarrow \blacklozenge(tryaccess(s,o,r) \wedge (p_1 \wedge \ldots \wedge p_i))$, were *s* is the subject, *o* is the object, and *r* is the right that the subject attempts to perform on the object.

UCON ***permitaccess*** correspond to the G-UCON action ***PermitUse***. In our one-player game – and as the ***System*** can enforce all steps alone –, the ***TransferPrivileges*** action can be ignored as the ***System*** can enforce the transition to the ***Delegating Privileges*** and then to the ***Requesting*** states like the following:

TransferPrivileges
$\langle\langle System \rangle\rangle \lozenge \left(CurrentState = Delegating\_privileges\right) \rightarrow$ $\langle\langle\ \rangle\rangle \blacklozenge \left(CurrentState = init\right)$

PermitDelegation
$\langle\langle System \rangle\rangle \lozenge \left(CurrentState = Requesting\right) \rightarrow$ $\langle\langle\ \rangle\rangle \blacklozenge \left(CurrentState = Delegating\_privileges\right)$

We capture the UCON ***permitaccess*** in the context of G-UCON ***PermitUse*** like the following:

PermitUse
$\bullet\ \langle\langle System \rangle\rangle \lozenge\ PermitUse\left(s, r(o)\right) \rightarrow$ $\langle\langle\ \rangle\rangle \blacklozenge \left(CurrentState = Requesting\ \wedge\ s.r \subseteq r(o).neededRights\ \wedge\ (p_1 \wedge \ldots \wedge p_i)\right)$

- $\langle\langle System\rangle\rangle\Diamond\big(CurrentState = Resource\_mapping\big) \rightarrow$

  $\langle\langle\ \rangle\rangle\blacklozenge\big(CurrentState = Requesting\ \wedge PermitUse(s, r(o))\big)$

where $p_1, \dots, p_i$ are predicates related to the G-UCON ***PermitUse*** action and correspond to the UCON ***tryaccess*** predicates, and *s, o*, and *r* have the same semantic like in UCON.

If we compare the last G-UCON policy with the UCON *preA₀* policy, we recognize how G-UCON can model UCON. The UCON *preA₀* policy reads:

$permitaccess(s,o,r) \rightarrow \blacklozenge(tryaccess(s,o,r) \wedge (p_1 \wedge \dots \wedge p_i))$,

while the relevant part of the corresponding G-UCON policy reads:

$$\langle\langle System\rangle\rangle\ \Diamond\ PermitUse\big(s, r(o)\big) \rightarrow$$
$$\langle\langle\ \rangle\rangle\blacklozenge\big(CurrentState = Requesting\ \wedge$$
$$\big(s.r \subseteq r(o).neededRights\big)\ \wedge\ (p_1 \wedge \dots \wedge p_i)\big).$$

In analogy, we are able to capture the UCON policies features by a suitable reduction of the G-UCON policies specification.

Example 2: Reducing G-UCON to UCON *preA₁*

Another example is the Usage Control policy for the model *preA₁* which reads:

$$permitaccess(s,o,r) \rightarrow$$
$$\blacklozenge(tryaccess(s,o,r) \wedge (p_1 \wedge \dots \wedge p_i)) \wedge \blacklozenge\, preupdate(attribute).$$

The corresponding G-UCON policy would be similar to the latter policy corresponding to *preA₀*, but with the *Requesting-update* added:

TransferPrivileges
$\langle\langle System\rangle\rangle\Diamond\ \big(CurrentState = Delegating\_privileges\big) \rightarrow$ $\langle\langle\ \rangle\rangle\blacklozenge\big(CurrentState = init\big)$

PermitDelegation

$$\langle\langle System\rangle\rangle\Diamond\left(CurrentState = Requesting\right) \rightarrow$$
$$\langle\langle\ \rangle\rangle\blacklozenge\left(CurrentState = Delegating\_privileges\right)$$

Requesting-update

- $\langle\langle System\rangle\rangle\Diamond\left(Requesting - update(attributes)\right) \rightarrow$
  $\langle\langle\ \rangle\rangle\blacklozenge\left(CurrentState = Requesting \ \wedge\ update(attributes) \ \wedge\right.$
  $\left.(p_1 \wedge \ldots \wedge p_i)\right)$

- $\langle\langle System\rangle\rangle\Diamond\left(CurrentState = Requesting\right) \rightarrow$
  $\langle\langle\ \rangle\rangle\blacklozenge\left(CurrentState = Requesting \ \wedge\ Requesting - update(attributes)\right)$

where *update(attributes)* is a function used to rewrite the attributes' values from their corresponding value space. For example:

$update(Book.maxAccess) : Book.maxAccess' = Book.maxAccess - 1$, or

$update(User.rights) : User.rights' = \{read\}$, or

$update(User.rights) : User.rights' = User.rights / \{write\}$.

PermitUse

- $\langle\langle System\rangle\rangle\Diamond\ PermitUse\left(s, r(o)\right) \rightarrow$
  $\langle\langle\ \rangle\rangle\blacklozenge\left(CurrentState = Requesting \ \wedge\ s.r \subseteq r(o).neededRights \ \wedge\right.$
  $\left.(p_1 \wedge \ldots \wedge p_i)\right)$

- $\langle\langle System\rangle\rangle\Diamond\left(CurrentState = Resource\_mapping\right) \rightarrow$
  $\langle\langle\ \rangle\rangle\blacklozenge\left(CurrentState = Requesting \ \wedge\ PermitUse\left(s, r(o)\right)\right)$

where $p_1, \ldots, p_i$ are related to the ***PermitUse*** predicates and correspond to UCON ***tryaccess*** predicates, and *s, o*, and *r* have the same semantic like in UCON.

Intuitively, it is possible to model the different UCON policies using G-UCON in analogy to the latter two examples. Beside the importance of describing other known access control models using G-UCON, the next subsection (5.2.2) deals with another important issue, namely what G-UCON can do more than the known access control models.

5.2.2. What can G-UCON do more than other access/usage control models?

That G-UCON is reducible to UCON makes G-UCON capable of handling the different authorization models that UCON can model, i.e. the traditional access control models including DAC, MAC, RBAC, Chinese Wall, etc. Different examples of how UCON can be reduced to these different models are included in the literature [23, 144-146]. A reduction from G-UCON to these models through UCON is possible. Anyhow, we limit ourselves in this work to the upper mentioned examples of modeling UCON *$preA_0$* and *$preA_1$* using G-UCON. Further reductions to traditional access control models (DAC, RBAC, etc.) are possible in two steps: reducing G-UCON to UCON and then reducing UCON to the specific traditional access control model. We will concentrate ourselves in this subsection on the new possibilities of expressing policies using G-UCON, which are not given using state-of-the-art models.

Applying G-UCON to define a usage policy allows inclusion of complex but still real situations like introducing new conditions during the using state. Another example is the immediate revoking of an ongoing usage according to a sudden threat or policy change by the VO manager, who does not have control on the used resources (see Figure 2, Figure 27, and Figure 28). The latter example is very realistic in the biomedical community, still anyhow not foreseen in the real world grid computing implementations. The only way is still to call the administrator of the used resource and ask her (after convincing her) to kill the particular UNIX job (see Figure 31). The following examples show similar and even more complex situations.

Example 3: Using G-UCON to define a suitable policy for research using genomic data in a grid computing environment

A researcher (subject) wants to access the genetic data (object) that belongs to a patient. The researcher (***User***) wants to perform correlation (***corrProcess***) between different datasets of different patients in order to identify the gene(s) responsible for a specific illness. For this purpose, the researcher defines her application to be a long term process: she will delegate the needed rights (***accessPatientGeneticData***) to the application (***corrApplication***), which will work on her behalf in order to run for a relatively long period (six months), and to search for any newly submitted genetic dataset from a patient, who has this illness. The G-UCON ***TransferPrivileges*** and ***PermitDelegation*** policies for this use case reads:

- $\langle\langle User\rangle\rangle\Diamond\, TransferPrivileges\big(Researcher, corrApplication, \{accessPatientGeneticData\}\big) \rightarrow \langle\langle\ \rangle\rangle\blacklozenge \Big(includeSubject\big(corrApplication, \{accessPatientGeneticData\}\big)\ \mathcal{U}\ \big(Denied \vee Revoked \vee End \vee corrApplication.runTime > 6\ months\big) \wedge (p_1 \wedge \ldots \wedge p_i)\Big)$

- $\langle\langle User\rangle\rangle\bigcirc\ \big(CurrentState = Delegating\_privileges\big) \rightarrow \langle\langle\ \rangle\rangle\circleddash\ \Big(CurrentState = init\ \wedge TransferPrivileges\big(Researcher, corrApplication, rights = \{accessPatientGeneticData\}\big)\Big)$

- $[\![\Sigma / User]\!]\bigcirc\ \big(CurrentState = Delegating\_privileges\big) \rightarrow \langle\langle\ \rangle\rangle\blacksquare\ \big(CurrentState = init\big)$

where $includeSubject(s,r) : Subjects' = Subjects \cup \{s\}\ \wedge\ s.rights = \{r\}$

- $\langle\langle User, UserSys\rangle\rangle\Diamond\ PermitDelegation\big(Researcher, corrApplication, \{accessPatientGeneticData\}\big) \rightarrow \langle\langle\ \rangle\rangle\blacklozenge\ \Big(\big(includeSubject(corrApplication, \{accessPatientGeneticData\})\ \mathcal{U}\ (Denied \vee Revoked \vee End \vee corrApplication.runTime = 6\ months)\big)\ \wedge\ (User.id = Researcher.id)\ \wedge\ (corrApplication \notin Subjects)\ \wedge\ \big(\{accessPatientGeneticData\} \subseteq User.rights\big)\ \wedge\ (p_1 \wedge \ldots \wedge p_i)\Big)$

- $\langle\langle User, UserSys\rangle\rangle\bigcirc\left(CurrentState = Requesting\right) \rightarrow$
  $\langle\langle\ \rangle\rangle\ominus\left(CurrentState = Delegating\_privileges \wedge PermitDelegation\left(Researcher, corrApplication, \{accessPatientGeneticData\}\right)\right)$

- $[\![\Sigma / \{User, UserSys\}]\!]\bigcirc\left(CurrentState = Requesting\right) \rightarrow$
  $\langle\langle\ \rangle\rangle\blacksquare\left(CurrentState = Delegating\_privileges\right)$

The patient signed a consent (***condition1***) that allows her physician to freely share her patient's genetic data with researchers under the condition that these data will be used only for this specific research (***condition2***) and that her data will never be stored on a machine, which belongs to her health insurance company (***condition3*** about ***ResourceSys***). Nevertheless, these data should be used in such a way, that the researcher cannot re-identify the patient (***condition4***).

- $\langle\langle User, BrokeringSys\rangle\rangle\Diamond\ PermitUse\left(corrApplication, corrProcess(patientData)\right) \rightarrow$
  $\langle\langle\ \rangle\rangle\blacklozenge\left(CurrentState = Requesting \wedge PermitDelegation(Researcher, corrApplication, rights) \wedge corrApplication.rights \subseteq corrProcess(patientData).neededRights \wedge (condition1 \wedge condition2 \wedge condition3 \wedge condition4 \wedge ... \wedge p_i)\right)$

- $\langle\langle User, BrokeringSys\rangle\rangle\bigcirc\left(CurrentState = Resource\_mapping\right) \rightarrow$
  $\langle\langle\ \rangle\rangle\ominus\left(CurrentState = Requesting \wedge PermitUse\left(corrApplication, corrProcess(patientData)\right)\right)$

- $[\![\Sigma / \{User, BrokeringSys\}]\!]\bigcirc\left(CurrentState = Resource\_mapping\right) \rightarrow$
  $\langle\langle\ \rangle\rangle\blacksquare\left(CurrentState = Requesting\right)$

where $condition3 \equiv \neg(ResourceSys = patients'\_health\_insurance\_company)$

While the interest of the researcher is to find the gene(s), her action could eventually harm the patient's privacy and lead to a re-identify of the patient (for example, a father and three of his children, who have the same illness). Anyhow, this disclosure risk is not to be anticipated, because we still have no knowledge about this risk when she started her correlation application (no indications that this illness is inheritable).

Normally in medical research networks, there is a security board[27] [294], which is responsible for supervising how the usage of the system as well as the usage of the data conforms to the legal requirements and/or ethical regulations. Such a security board has a direct access to enforce the policy of the ***BrokeringSys***. The security board may introduce a new condition (***newCondition***) during the usage. At this point, we can realize the importance of the continuity of decision aspects in G-UCON (see Figure 27). After a period of three months the latter ***permitUse*** policy may be changed to become:

- $\langle\langle User, BrokeringSys\rangle\rangle \Diamond\, PermitUse\big(corrApplication, corrProcess(patientData)\big) \rightarrow$
  $\langle\langle\ \rangle\rangle \blacklozenge \big(CurrentState = Requesting \ \wedge$
  $PermitDelegation\big(Researcher, corrApplication, rights\big) \ \wedge$
  $corrApplication.rights \subseteq corrProcess(patientData).neededRights \wedge$
  $(condition1 \wedge condition2 \wedge condition3 \wedge condition4 \wedge$
  $newCondition \wedge ... \wedge p_i)\big)$

- $\langle\langle User, BrokeringSys\rangle\rangle \bigcirc \big(CurrentState = Resource\_mapping\big) \rightarrow$
  $\langle\langle\ \rangle\rangle \ominus \big(CurrentState = Requesting \ \wedge$
  $PermitUse\big(corrApplication, corrProcess(patientData)\big)\big)$

- $[\![\Sigma / \{User, BrokeringSys\}]\!] \bigcirc \big(CurrentState = Resource\_mapping\big) \rightarrow$
  $\langle\langle\ \rangle\rangle \blacksquare \big(CurrentState = Requesting\big)$

[27] The MediGRID Security Board was introduced in the MediGRID Resource Usage Policy – phase one.

A resource provider (***ResourceSys1***) has a strategy to not allow a process from the class ***corrProcess*** to run more than one hour on his resources, while another resource provider (***ResourceSys2***) do not have such a limit. Nevertheless, ***ResourceSys1*** has better CPUs (e.g. PIV) and faster connection to the patient databanks than ***ResourceSys2*** (e.g. PIII). Here the ***Researcher*** develops a strategy; i.e. a winning strategy: whenever it is possible, the work should be performed using the ***ResourceSys1*** resources, otherwise using the ***ResourceSys2*** resources. ***BrokeringSys*** does not have any policy regarding this.

- $\langle\langle ResourceSys1\rangle\rangle\Diamond\ PermitLogin1\big(corrApplication, corrProcess.aresource, resource\big)\rightarrow$
  $\langle\langle\ \rangle\rangle\blacklozenge\big(CurrentState = Resource\_mapping\ \wedge$
  $PermitUse(corrApplication, corrProcess(patientData)\ \wedge$
  $corrProcess.aresource = resource\ \wedge$
  $(\ corrProcess.runTime < \ 1\ hour\ \wedge p_2 \wedge ... \wedge p_i)\big)$

- $\langle\langle ResourceSys2\rangle\rangle\Diamond\ PermitLogin2\big(corrApplication, corrProcess.aresource, resource\big)\rightarrow$
  $\langle\langle\ \rangle\rangle\blacklozenge\big(CurrentState = Resource\_mapping\ \wedge$
  $PermitUse\big(corrApplication, corrProcess(patientData)\big)\ \wedge$
  $corrProcess.aresource = resource\ \wedge\ (\ p_1 \wedge ... \wedge p_i)\big)$

- $\langle\langle User\rangle\rangle\Diamond\ PermitLogin\big(corrApplication, corrProcess.aresource, resource\big)\rightarrow$
  $\big(\langle\langle\ \rangle\rangle\Box\Diamond PermitLogin1\ \vee \langle\langle\ \rangle\rangle\Box\Diamond\big(\neg PermitLogin1 \wedge PermitLogin2\big)\big)$

- $\big(\langle\langle User, ResourceSys1\rangle\rangle\bigcirc\big(CurrentState = User\_mapping\big)\ \vee$
  $\langle\langle User, ResourceSys2\rangle\rangle\bigcirc\big(CurrentState = User\_mapping\big)\big)\rightarrow$
  $\langle\langle\ \rangle\rangle\ominus\ \big(CurrentState = Resource\_mapping\ \wedge$
  $PermitLogin\big(corrApplication, corrProcess.aresource, resource\big)\big)$

- $[\![\Sigma / \{User, ResourceSys1, ResourceSys2\}]\!]\bigcirc\ \big(CurrentState = User\_mapping\big)\rightarrow$
  $\langle\langle\ \rangle\rangle\blacksquare\ \big(CurrentState = Resource\_mapping\big)$

Now the ***ResourceSys1*** always map processes like ***corrProcess*** to a special local user account (***corrProcessUser***), which fulfills the legal framework of these processes (like encrypted storing of the genetic data). The ***BrokeringSys***, anyhow, demands that this local account should be newly created in order to guarantee that no interfering with other accounts could happen (***corrProcessUser.logins***). The Researcher demands that all information stored should be encrypted with 2048 bit (***corrProcessUser.encryptingKeyLong***) :

- $\langle\langle User, BrokeringSys, ResourceSys\rangle\rangle\Diamond$
  $PermitTask\big(corrApplication, corrProcessUser, corrProcess(patientData)\big) \rightarrow$
  $\langle\langle\,\rangle\rangle\blacklozenge\big(CurrentState = User\_mapping \;\wedge$
  $PermitLogin\big(corrApplication, corrProcess.aresource, resource\big) \;\wedge$
  $(corrProcessUser.logins = 0 \wedge$
  $corrProcessUser.encryptingKeyLong = 2048 \;\wedge\; p_3 \wedge ... \wedge p_i)\big)$

- $\langle\langle User, BrokeringSys, ResourceSys\rangle\rangle\bigcirc\,(CurrentState = Using) \rightarrow$
  $\langle\langle\,\rangle\rangle\ominus\,(\,CurrentState = User\_mapping \;\wedge$
  $PermitTask(corrApplication, corrProcessUser, corrProcess(patientData))\,)$

- $[\![\Sigma / \{User, BrokeringSys, ResourceSys\}]\!]\bigcirc\,\big(CurrentState = Using\big) \rightarrow$
  $\langle\langle\,\rangle\rangle\blacksquare\,\big(CurrentState = User\_mapping\big)$

The researcher wants to end her ***corrProcess*** as soon as she has a positive result (for example, positive result could mean 20 identical SNPs in two different genomic series and then the data from both patients should enter a second analysis level):

- $\langle\langle User\rangle\rangle\Diamond\; EndUse\big(corrApplication, corrProcessUser,$
  $corrProcess(patientData)\big) \rightarrow$
  $\langle\langle\,\rangle\rangle\blacklozenge\,\big(CurrentState = Using \;\wedge$
  $PermitTask\big(corrApplication, corrProcessUser,$
  $corrProcess(patientData)\big) \wedge$
  $(corrProcess.foundSNPs \geq 20 \wedge p_2 \wedge ... \wedge p_i)\big)$

- $\langle\langle User\rangle\rangle \bigcirc \big(CurrentState = End\big) \rightarrow \langle\langle\ \rangle\rangle \ominus \big(CurrentState = Using \wedge EndUse(corrApplication, corrProcessUser, corrProcess(patientData))\big)$

- $[\![\Sigma / User]\!] \bigcirc \big(CurrentState = End\big) \rightarrow \langle\langle\ \rangle\rangle \blacksquare \big(CurrentState = Using\big)$

Example 4: Enforcing the re-identification risk control using G-UCON to maintain k-anonymity

K-anonymity [295] is one of the main requirements that is used by agencies releasing data. K-anonymity is a measure to enforce that any released data entry is indistinguishably related to no less than a certain number of entries [296]. The goal is to keep the probability of a positive re-identification of a subject within acceptable limits. We can conceive k-anonymity be means of two tables: de-identified medial table (the data to be released), and the non de-identified publicly available table (see Table 9 and Table 10).

Table 9: Example of de-identified medical data

Name	Address	ZIP	Date of Birth	Gender	Ethnicity	Visit date	Diagnoses
		37075	29.05.1984	M	White	01.03.2008	Chest pain
		37080	16.07.1975	M	White	05.02.2008	Brocken arm
		37075	07.02.1977	F	Asian	17.03.2008	Brocken arm
		37075	08.10.1970	F	Asian	24.02.2008	Stomach pain
		37073	18.06.1972	M	White	22.04.2008	Stomach pain
		37084	23.09.1978	F	White	22.04.2008	Chest pain
		39343	16.09.1968	M	Black	13.03.2008	Headache
		39120	22.07.1970	F	White	19.02.2008	Stomach pain

Table 10: Example of non de-identified publicly available data

Name	Address	ZIP	Date of Birth	Sex	Marital Status
…	…	…	…	…	…
Lin Fu	Zimmermannstr. 5	37075	07.02.1977	F	Single
…	…	…	…	…	…

The set of attributes included in both tables are called quasi-identifiers. Quasi-identifiers are exploitable for linking, which could lead to re-identification of subjects/patients (see Figure 32). K-anonymity requirement means that "each release of data must be such that every combination of values of quasi-identifiers can be indistinctly matched to at least k respondents" [296]. Because it is impractical to make assumptions on the publicly available datasets, the k-anonymity concept requires that each quasi-identifier value in the released table has at least k occurrences [295, 296]. Merging and correlating different datasets of the same subjects is becoming more important for medical application, e.g. Genome Wide Association Studies [257, 258]. In such cases, the service provider maintains the k-anonymity by different methods, which are beyond the scope of this work and can be found in literature [28, 295-298].

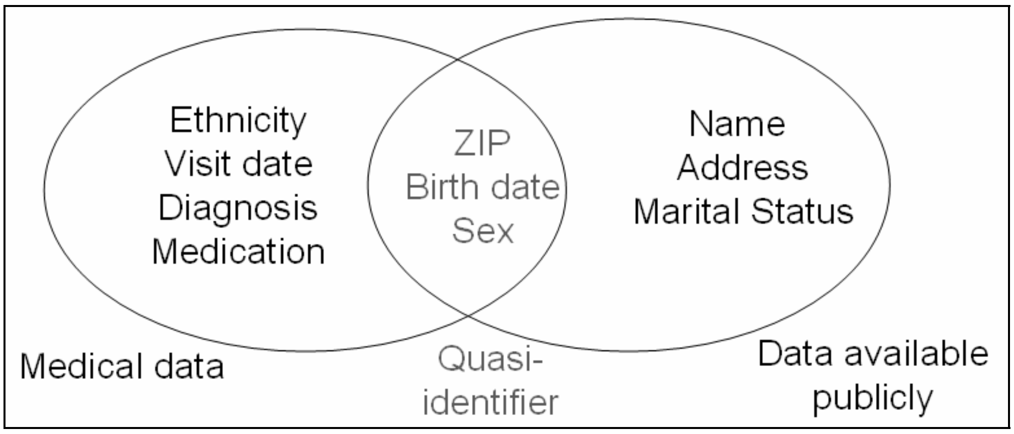


Figure 32: Quasi-identifiers shared in two datasets.

Considering the normal use case of having a user and a service provider (not a grid computing environment), the authorization system of the service provider has to enforce k-anonymity on all released data. Therefore, the service provider will normally maintain lists of all users and their access history in order to enforce the k value on each new access request (see Figure 33). The Chinese Wall security model was developed for such use cases [77]. Using the attributes mutability feature introduced in UCON, we can also enforce such a policy. In temporal logic and for k value of 100, we would refer to such a condition with $\Box(k > 100)$ – it is always the case that k > 100. Such a condition can be easily considered in an appropriate UCON policy like *preA*$_1$.

In the real world applications, conflict of interest has been reported: the user does not want the service provider to know about her research and to store a profile of the data used by her (intellectual property rights of the user). At the same time the service provider aim is to enforce the required k value. Such conflict situation were indeed reported as being problematic cases [64] (see subsection 4.1.1). In a grid environment, such problem is more serious. What makes the grid a suitable environment for a re-identification is that the service providers will not or are not allowed to share the users' profiles between themselves. This gives the user the possibility to access different datasets which include information on the same subject from different providers. The reports about the disclosure risk in grid computing environments warn that gird computing is highly disclosive environments [11, 203].

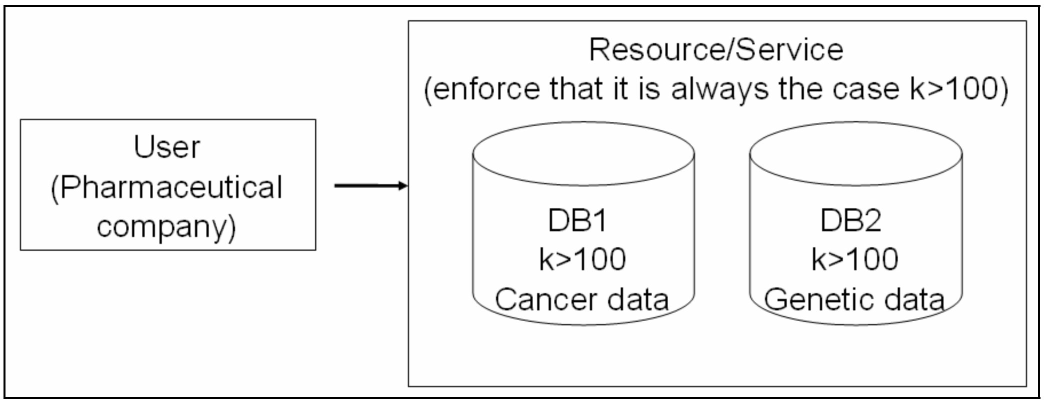


Figure 33: Re-Identification risk control in a normal use case like offering a direct access to the data by means of a web service (non-grid environment). State-of-art authorization models like Chinese Wall or UCON can model this situation.

The solution that G-UCON offers is the modeling of a multi-agent system with strict linking of the policy enforcement point to the corresponding agent (entity). A corresponding entity could be a third party like the resource mapping agent, who is trusted from both: the user and the service provider. The point here is to produce a flexibility to transfer the policy enforcement point between the different entities and not to keep it by the service provider. As G-UCON uses a multi-agent approach to define a policy, this allows the different agents to cooperate and to work on fulfilling the needs of other agents. G-UCON allows us to enforce the service provider policy by the resource mapping agent (***BrokeringSys***) in order not to harm the intellectual property rights of the user, as this mapping agent is trusted by both (see Figure 34).

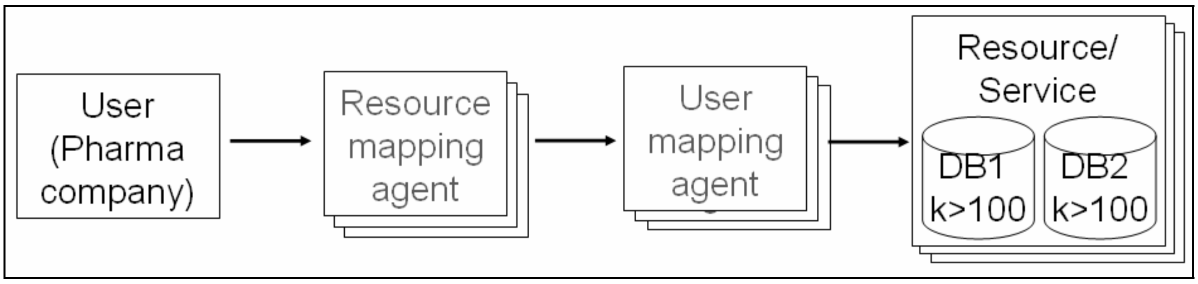


Figure 34: A use scenario which requires re-Identification risk control in a grid environment. G-UCON can model this situation.

Such a policy can be considered within the ***DenyUse*** policy, which would be like the following:

- $\langle\langle BrokeringSys\rangle\rangle\Diamond\ DenyUse\big(subject, process(object)\big)\rightarrow$
  $\langle\langle\ \rangle\rangle\blacklozenge\ \big(CurrentState = Requesting\ \wedge\ (k < 100 \wedge p_2 \wedge \ldots \wedge p_i)\big)$

- $\langle\langle BrokeringSys\rangle\rangle\bigcirc\big(CurrentState = Denied\big)\ \rightarrow$
  $\langle\langle\ \rangle\rangle\ominus\ \big(CurrentState = Requesting\ \wedge DenyUse\big(subject, process(object)\big)\big)$

- $[\![\Sigma / \{BrokeringSys\}]\!]\bigcirc\big(CurrentState = Denied\big)\ \rightarrow$
  $\langle\langle\ \rangle\rangle\blacksquare\ \big(CurrentState = Requesting\big)$

Example 5: the D-Grid experience

One aim of D-Grid is to connect the computer centers in Germany with each other. Anyhow, the different computer centers have different policies for the users. While most computer centers allow any researcher participating in D-Grid to access the grid resources, some do not. The Leibniz computer center (LRZ) in Munich does not allow users belong of special nationalities to access its resources; at present "these are in particular residents of the following countries: Cuba, Iran, Iraq, Libya, North Korea, the Sudan and Syria" [299]. While other D-Grid resources centers do not have such regulation (e.g. the Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen – GWDG), LRZ has to enforce this extra policy due to governmental regulations [299]. This is an example of how the different policies of the regional computer centers are influenced by local political regulations.

An eventual D-Grid user from one of the mentioned countries has to enforce by herself that the submitted jobs will not be sent to LRZ. For the time being, this is possible by performing the job submission using the command line. Automatized job submission mechanisms, for instance Grid Workflow Execution Service – GWES[28] [300], do not and cannot take into account different policies of the different

[28] Currently all MediGRID services use GWES, which makes these services more reliable and simple for the end user.

participating parties. For example, GWES will simply send the GT4 jobs automatically to any available free resource without considering such special policies.

Normally, for the users it does not matter where to compute their jobs, rather it is important to have the result as soon as possible. A delay may last hours because of long jobs queue. We can imagine of D-Grid use scenarios, in which the user has to wait hours to learn that her job is not executed because a scheduling service forwarded her job (jobs) to a resource, which does not accept her nationality.

Within the framework of G-UCON as a model for authorization, it is required to define explicitly which entity has to perform what responsibility. By an accurate resource mapping process, the scheduling service can perform a correct match from abstract resources to available real resources, which can accept the user's request. While this is intuitively required, current scheduling is aiming at finding available resources, rather than finding available resources that authorize the user request. The user is authorized to use grid resources, but not to use all the available grid resources. The first attempts to grid authorization services failed to capture such use scenarios [120, 126].

To put it simple, in the original grid computing vision [32, 35], it is the responsibility of the system to find out where the user can submit her job and not the user's responsibility, i.e. it is the responsibility of the authorization system to find out where the user's request can be authorized and not the user's duty. With G-UCON we can model such a situation, in which the pre-requests of the different providers are already considered. Recalling the specification of G-UCON (see subsection 4.3.1 and Figure 29), the context of the ***DenyLogin2*** action performed by the user mapping agent (the ***ResourceSys*** which is the LRZ in this example) can explicitly capture such use scenarios:

- $$\langle\langle LRZ\rangle\rangle\Diamond\ DenyLogin2\big(subject, process.aresource, resource\big)\rightarrow \\ \langle\langle\ \rangle\rangle\blacklozenge\ \big(CurrentState = Resource\_mapping\ \wedge \\ (subject.nationality \in \{Cuba, Iran...\} \wedge p_2 \wedge ... \wedge p_i)\big)$$

- $\langle\langle LRZ \rangle\rangle \bigcirc (CurrentState = Requesting) \rightarrow$
  $\langle\langle\ \rangle\rangle \ominus (CurrentState = Resource\_mapping \wedge$
  $DenyLogin2(subject, process.aresource, resource))$

- $[\![\Sigma / LRZ]\!] \bigcirc (CurrentState = Requesting) \rightarrow$
  $\langle\langle\ \rangle\rangle \blacksquare (CurrentState = Resource\_mapping)$

The aim is actually not to send the job to a resources center, which as we already know will not accept this job. Therefore the resource mapping agent (which is the ***BrokeringSys*** or GWES in the case of MediGRID) could capture this in the specification of the ***PermitUse*** action.

- $\langle\langle User, BrokeringSys \rangle\rangle \lozenge\ PermitUse(subject, process(object)) \rightarrow$
  $\langle\langle\ \rangle\rangle \blacklozenge (CurrentState = Requesting \wedge$
  $(subject.id = User.id \vee PermitDelegation(User, subject, rights)) \wedge$
  $subject.rights \subseteq process(object).neededRights \wedge$
  $((ResourceSys \notin \{LRZ\} \wedge User.nationality \in \{Cuba, Iran, ...\}) \wedge$
  $p_2 \wedge ... \wedge p_i))$

- $\langle\langle User, BrokeringSys \rangle\rangle \bigcirc (CurrentState = Resource\_mapping) \rightarrow$
  $\langle\langle\ \rangle\rangle \ominus (CurrentState = Requesting \wedge$
  $PermitUse(subject, process(object)))$

- $[\![\Sigma / \{User, BrokeringSys\}]\!] \bigcirc (CurrentState = Resource\_mapping) \rightarrow$
  $\langle\langle\ \rangle\rangle \blacksquare (CurrentState = Requesting)$

With such a ***PermitUse*** policy we can avoid in advance a later rejection by the targeted resources because of special regulations of the resources centers. It is maybe difficult to see the novelty of the latter ***PermitUse*** policy. Therefore, this aspect is emphasized here.

It is obvious that we send jobs to a specific resource only if we have access rights to it. Languages like XACML, frameworks like the policy enforcement points, and policy decision points are already in use since years [301, 302]. What is new in G-UCON in this context is the authorization role of the brokering system. The brokering system in G-UCON can and has to enforce the policy of the user as well as its own policy in order to reach a maximum output, i.e. for example no delay in executing jobs on the resources. In stead of reporting denied or granted access/use of a specific resource, the brokering system in G-UCON has to find on which resource the user’s job can be performed and is authorized to run. Current brokering systems are considered till now just as a job-scheduling or a job-brokering functionality, i.e. to find the needed resource for the job (e.g. GWES), but not as a user-brokering functionality, i.e. to search for the suitable resource for the user as well as her job. As the G-UCON framework considers the mapping agents as entities that affect the authorization process, finding the suitable resource for the user along side with her job can be considered. A grid system structure and mechanisms that consider G-UCON authorization model will be aware of this problem and can solve it.

Example 6: The MediGRID experiences

From the beginning, the different D-Grid community projects have different resources providers located all over Germany. In a required resources expansion step, the different communities obtained in 2006 and 2007 special funding in order to acquire their own resources as well as resources centers. By the end of the year 2007, there were the MediGRID resources, the InGrid resources, the AstroGrid resources and so on. As all resources were funded in the D-Grid initiative framework, the different communities can use them all mutually whenever it is necessary and possible. It was also requested from each community member to use primarily their community resources. That is: a MediGRID user should use the MediGRID resources, unless there are no free MediGRID resources she can use free resources from other D-Grid communities.

During the operation of the MediGRID resources, the e-science module, as the responsible for the scientific operation of MediGRID, observed the problem that some MediGRID users did not hold to this policy. Users from MediGRID used resources from other communities although there were free MediGRID resources.

The source of this problem was conflict of interest, which can be demonstrated as followed. The MediGRID VO management entity wants to enforce that all participating users have to use primarily MediGRID resources according to the MediGRID policies. Some MediGRID users prefer to use resources from other communities, because of special interest like more comfortable job submission regardless the MediGRID strict usage policy.

As MediGRID uses the conventional grid computing technology, it was not possible to enforce the MediGRID policy. The user continues to use the resources from other communities and there was no way to manage the members of the MediGRID VO correctly (see Figure 35).

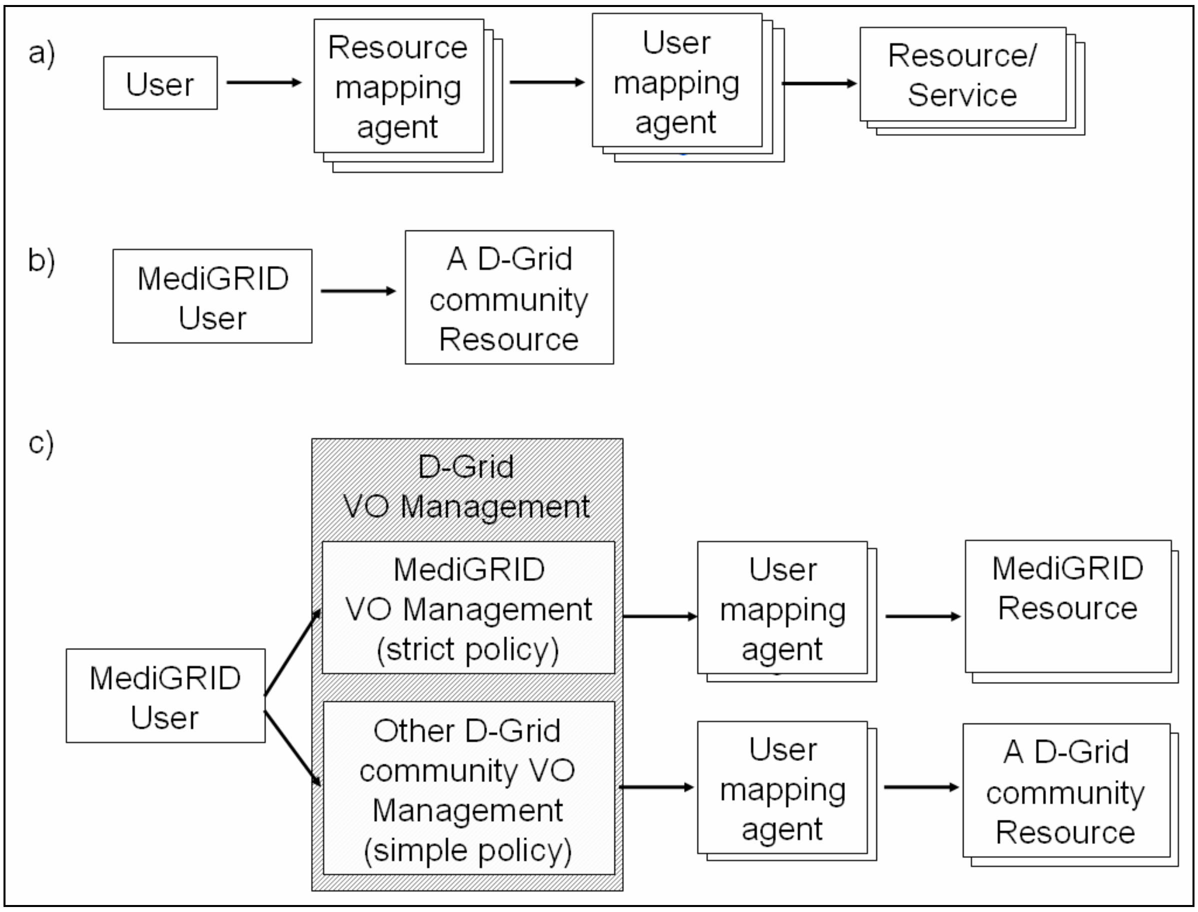


Figure 35: A problem appeared in the MediGRID project. a) The same as in Figure 2: A simplification of the resources access process in grids. b) The situation resulted from the fact that some MediGRID users started to use resources from other community because of the strict MediGRID policy. c) The MediGRID VO management could not enforce the MediGRID policy on the users because used mechanisms do not consider the normal grid authorization process.

As the G-UCON model considers VO management entities (i.e. brokering entities), it can model a similar situation like the one appeared in MediGRID. The solution would be simple by using the context of ***PermitUse***. What G-UCON can naturally model is that the authorization process can only take place via cooperation between the user and her VO management entity (VOM). Otherwise, the VOM entity can alone deny the use by performing ***DenyUse*** action, in case that this cooperation is not successful.

- $\langle\langle MediGridUser, MediGRID\_VOM \rangle\rangle \lozenge$
  $PermitUse\big(subject, process(object)\big) \rightarrow$
  $\langle\langle\ \rangle\rangle \blacklozenge \big(CurrentState = Requesting\ \wedge$
  $\big(subject.id = User.id \vee PermitDelegation(User, subject, rights)\big)\ \wedge$
  $subject.rights \subseteq process(object).neededRights\ \wedge$
  $(ResourceSys \in \{MediGRID\_Resources\} \wedge p_2 \wedge \ldots \wedge p_i)\big)$

- $\langle\langle MediGridUser, MediGRID\_VOM \rangle\rangle \bigcirc$
  $\big(CurrentState = Resource\_mapping\big)\ \rightarrow$
  $\langle\langle\ \rangle\rangle \ominus \big(CurrentState = Requesting\ \wedge$
  $PermitUse\big(subject, process(object)\big)\big)$

- $[\![\Sigma / \{MediGridUser, MediGRID\_VOM\}]\!] \bigcirc$
  $\big(CurrentState = Resource\_mapping\big)\ \rightarrow$
  $\langle\langle\ \rangle\rangle \blacksquare \big(CurrentState = Requesting\big)$

And the corresponding ***DenyUse*** action would look like the following:

- $\langle\langle MediGRID\_VOM \rangle\rangle \lozenge\ DenyUse\big(subject, process(object)\big) \rightarrow$
  $\langle\langle\ \rangle\rangle \blacklozenge\ \big(CurrentState = Requesting\ \wedge$
  $(ResourceSys \notin \{MediGRID\_Resources\} \wedge p_2 \wedge \ldots \wedge p_i)\big)$

- $\langle\langle MediGRID\_VOM \rangle\rangle \bigcirc \big(CurrentState = Denied\big) \rightarrow$
  $\langle\langle\ \rangle\rangle \ominus \big(CurrentState = Requesting \ \wedge$
  $DenyUse\big(subject, process(object)\big)\big)$

- $[\![\Sigma / \{MediGRID\_VOM\}]\!] \bigcirc \big(CurrentState = Denied\big) \rightarrow$
  $\langle\langle\ \rangle\rangle \blacksquare \big(CurrentState = Requesting\big)$

To enforce the last two policies means that we need to develop the suitable mechanisms to perform this enforcement. This is part of the designing objectives of a suitable authorization model for grid computing environments; such a model has to underline what exactly should be developed and extended in the lower levels of the security system (see Figure 3 as well as Figure 36).

Example 7: G-UCON modeling in non-grid situations – using G-UCON to model a complex UCON authorization example

In order to demonstrate the powerfulness of UCON, Sandhu et al. used the following example [23, 144]: a doctor (s) can perform (r) a particular operation (o). This can be expressed using the UCON *preA₀* model. A junior doctor can perform an operation only when there is a senior doctor monitoring the operation. An ongoing obligation $ob\_monitor(s1, s2)$ is defined where *s1* is the obligation subject and *s2* is the obligation object. This model is a combination of *preA₀* and *onB₀*. The UCON policy reads:

- $permitaccess(s, o, operate) \rightarrow$
  $\blacklozenge\,(\,tryaccess(s, o, operate) \wedge \ s.role = \ junior\_doctor))$

- $\square(\neg((s1.role = senior\_doctor) \wedge (ob\_monitor(s1, s)) \wedge$
  $(state(s, o, operate) = accessing) \rightarrow revokeaccess(s, o, operate)$

Considering that a senior doctor can allow an operation only if she is personally in the operation room, the important part of the corresponding G-UCON policy would be:

- $\langle\langle s\rangle\rangle\Diamond\ PermitUse(s1, operate(o)) \rightarrow$
  $\langle\langle\ \rangle\rangle\blacklozenge\left(CurrentState = Requesting\ \wedge\ s1.role \supseteq r(o).neededRole\ \wedge\right.$
  $\left. s.role = monitor.neededRole\right)$

where $r(o).neededRole = \{junior\_doctor,\ senior\_doctor\}$ and
$monitor.neededRole = \{senior\_docthor\}$

Now suppose that we want to allow the senior doctor to be able to overtake the operation, whenever she feels that the junior doctor is performing a mistake[29]:

- $\langle\langle s\rangle\rangle\Diamond\ RevokeUse(s1, operate(o))\ \rightarrow$
  $\langle\langle\ \rangle\rangle\blacklozenge\ (CurrentState = Using\ \wedge\ perform\_mistake(s1))$

- $\langle\langle s\rangle\rangle\bigcirc(CurrentState = Revoked)\ \rightarrow$
  $\langle\langle\ \rangle\rangle\ominus(CurrentState = Using\ \wedge RevokeUse(s1, operate(o)))$

- $[\![s1]\!]\bigcirc(CurrentState = Revoked)\ \rightarrow\langle\langle\ \rangle\rangle\blacksquare\ (CurrentState = Using)$

The latter situation – of overtaking the operation by a senior doctor – can not be modeled using UCON. UCON can indeed model that whenever the junior doctor performs a mistake, the use should be revoked. But using G-UCON, we can model that the judgment that the junior doctor is performing a mistake is controlled by the senior doctor. While state-of-the-art authorization models leave this judgment to the system, with G-UCON we can define explicitly who has the right to perform this judgment as well as who has to follow this judgment.

[29] We simplify the syntax of *RevokeUse* by omitting the context of (*localAccount*), which is not needed here. This simplification is possible by ignoring the context of mapping agents in G-UCON, which is also not needed here.

5.3. Outlook on G-UCON

Authorization in grid computing environments is still a problem because the used approaches designed originally for non-grid computing environments. The current approaches follow a down-top adaptation of the technology, in which an off-the-shelf authorization approaches are used and modification/extension are added to suite the grid computing environment. The approach used in this work differs in that it finds the origin of the problem first, which is the lack of a suitable model for authorization and then tackle the problem. The performed analysis resulted in the need to use a filling-the-gab approach rather than to apply a down-top adoption of the available technological solutions. The effort of this work is to fill this gab and yielded the G-UOCN authorization model. The different discussed scenarios demonstrate how G-UCON is capable of modeling the authorization process in a grid environment.

G-UCON is meant to be a first step in designing a solution for enforcing authorization in a grid computing environment. It allows us to describe the needed functionalities, whether they exist (in the mechanisms and architecture level – see Figure 36) or not. In this regard, specific needed add-ons to fulfill particular functionalities can be identified using G-UCON. The model serves the need to move stepwise from the requirements to the implementation. This refinement process from a higher level of abstraction to a more detailed description is a known method when dealing with complex problems. This stepwise move from the “what” to the “how” question (see Figure 3) is essential and correspond to the move from information to knowledge in the Ackoff’s wisdom model [165] (see section 3.2). This is the case when developing a solution for a complex problem and in designing and finding the characteristics of a complex system like the grid or like the authorizing problem in the grid [35]. For security engineering, this was proposed as a layered approach in [45]. In this layered architecture, G-UCON is an authorization model.

G-UCON needs extension of the standard grid architecture and mechanisms

Following the OM-AM (Objectives, Models, Architectures, and Mechanisms) framework for security engineering [45], we still need suitable architectures and suitable mechanisms to be able to enforce a policy written using G-UCON (see

Figure 36). That means, we have to verify that policies written using G-UCON can be mapped to the grid computing architecture as well as the grid mechanisms level; i.e. Open Grid Service Architecture (OGSA) [40, 55], WSRF [169, 222], XACML [302], SAML, Globus, etc. Extensions of the architectures and the mechanisms seem to be needed. For example, we cannot implement an obligation framework because of missing needed mechanisms to enforce this notion, we do not have mechanisms to implement all the aspects of multi-agency, and we have neither mechanisms nor suitable architecture to realize the temporal aspect of G-UCON. Supplemental mechanisms and architecture components are required. Although this is commonly known and can be nearly found in each presentation or article about grid computing security, G-UCON provides the possibility to identify these needed supplements. What exact components are needed is a specific issue related to the target application. A detailed discussion about possible supplements according to the application is left for future work. This discussion would be driven chiefly by the used data and the security level needed for the application (see Table 6).

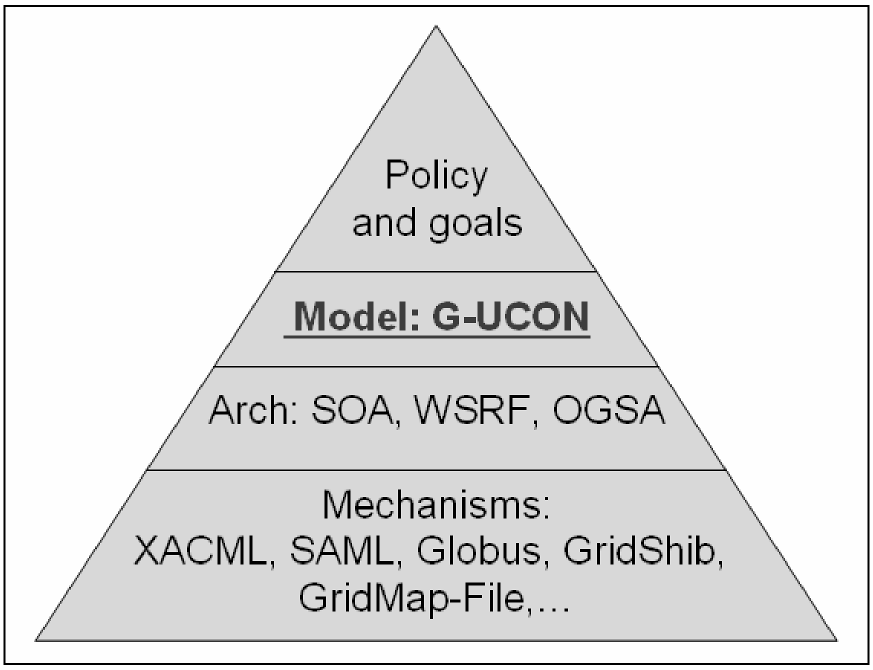


Figure 36: The OM-AM framework [45] for G-UCON. G-UCON is a model to develop and check policies for the usage of grid resources.

Modeling the statefulness and statelessness notions

G-UCON has four building blocks. A main building block is the UCON authorization model. One critique about UCON is that it does not include the possibility to model sessions [303]. This is also inherited to G-UCON. This topic is not handled in this work because grid services themselves are stateful service [169, 222]; i.e. they establish sessions in nature. The meaning and the relation between stateful services and sessions are beyond this work and can be found in the literature [304]. Whether we need to model sessions in the model level or it is sufficient to use them in the architecture level (see Figure 36) is a question that needs a deeper study in the future. Precisely: can we map the sessions' concept with all its properties in the grid architecture to the authorization model level, i.e. to G-UCON level or not? At this moment, we are only aware about this possible problem. A verifying if other such mismatches do exist is again an aspect for deeper and future study.

Actual vs. target validation of G-UCON

G-UCON is written using quasi-mathematical formulas. Because of its multi-agent nature, G-UCON uses ATL on concurrent game structures to describe its specifications. These formulas demonstrate how the authorization model for grid computing environments should look like and are not yet verified. Verifying the validity of the G-UCON formulas is currently not performed because the lack of a suitable validation tool. While this is a very important step towards the use of this model to develop a functional grid authorization system, this step is left for future work and a validation of the concept through different examples is provided. Most access control models were verified in the beginning in a similar way [23-25, 71, 74, 75, 81]. This leaves a complete mathematical validation for future work. Anyhow, before being able to validate G-UCON, we need:

- to develop a new temporal logic on top of ATL that can better capture the notion of creation/deletion of agents as well as grid delegation procedure,
- to develop a transition system that matches to this logic (it would be an extension of the concurrent game structure introduced by Alur et al. for ATL), and
- to implement a suitable model checker for this transition system as well as for the suitable temporal logic.

G-UCON in the grid computing authorization landscape

Developing an authorization model for a modern computing system is not the work of one person or one group. Currently used authorization systems and models were developed over years by different groups and institutions. Through this vital development, the concept of authorization itself has been evolved. In series of different efforts conducted by multiple groups analyzing the grid security problem, the analysis identifies a deficiency of a suitable authorization model, which hinders a transfer of the grid computing technology into the medical discipline. This thesis tackles this deficiency by developing a new authorization model – the Grid Usage Control model (G-UCON), which is a first step to fill the gap. G-UCON – as a whole or partially – will be of a benefit for the transfer of the grid computing technology to the medical discipline. The model may inspire the design of more suitable authorization systems in the future.

Abbreviations

ACL	Access Control Lists
ANSI	American National Standards Institute
ASM	Abstract State Machine
ATL	Alternating-Time Temporal Logic
BMBF	German Federal Ministry of Education and Research
CA	Certification Authority
caBIG	Cancer Biomedical Informatics Grid
CAS	Community Authorization Service
CORBA	Common Object Request Broker Architecture
CRM	Client-side Reference Monitor
CTL	Computational Tree Logic
DAC	Discretionary Access Control
DRM	Digital Right Management
DRS	Data Replication Service
GRAM	Globus Resource Allocation Manager
GSI	Grid Security Infrastructure
G-UCON	Grid Usage Control
GT	Globus Toolkit
GWES	Grid Workflow Execution Service
HL7	Health Level Seven
HIPAA	Health Insurance Portability and Accountability Act
INCITS	International Committee for Information Technology Standards
IP	Internet Protocol
IPR	Intellectual Property Rights
ISO	International Standards Organization
LTL	Linear Temporal Logic
MAC	Mandatory Access Control
MDS	Monitoring and Discovery System
MLS	Multilevel Security Systems
Non-DAC	Non-Discretionary Access Control
OGSA	Open Grid Service Architecture
OGSA-DAI	OGSA - Data Access and Integration Service
OM-AM	Objectives, Models, Architectures, and Mechanisms framework

OSI	Open Systems Interconnection
PERMIS	Privilege and Role Management Infrastructure Standards
PET	Privacy Enhancing Technologies
PKI	Public Key Infrastructure
PRIMA	Privilege Management and Authorization
RBAC	Role Based Access Control
RFC	Request for Comments
RFT	Reliable File Transfer
RLS	Replica Location Services
SAML	Security Assertion Markup Language
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SNP	Single Nucleotide Polymorphism
SRB	Storage Resource Broker
SRM	Server-side Reference Monitor
SSL	Secure Socket Layer
TCSEC	Trusted Computer System Evaluation Criteria – the Orange Book
TLA	Temporal Logic of Action
UCON	Usage Control
VC	Virtual Corporation
VM	Virtual Machine
VO	Virtual Organization
VOMS	Virtual Organization Management Service
WS	Web Services
WSRF	Web Services Resource Framework
XACML	Extensible Access Control Markup Language
XML	Extensible Markup Language
XrML	Extensible Rights Markup Language

References

[1] Anderson, R., *Security engineering: a guide to building dependable distributed systems*. 2001, New York: Wiley. xxviii, 612 p.

[2] Anderson, R., Isaac Newton Institute for Mathematical Sciences, and British Medical Association, *Personal medical information: security, engineering, and ethics, Proceedings of the Personal Information Workshop, Cambridge, U.K., June 21-22, 1996*. 1997, Berlin; New York: Springer. x, 250 p.

[3] Martin-Sanchez, F., V. Maojo, and G. Lopez-Campos, *Integrating genomics into health information systems.* Methods of Information in Medicine, 2002. **41**(1): p. 25-30.

[4] Diero, L., et al., *Can data from an electronic medical record identify which patients with pneumonia have Pneumocystis carinii Infection?* International Journal of Medical Informatics, 2004. **73**(11-12): p. 743-750.

[5] Haraldsson, H., L. Edenbrandt, and M. Ohlsson, *Detecting acute myocardial infarction in the 12-lead ECG using Hermite expansions and neural networks.* Artificial Intelligence in Medicine, 2004. **32**(2): p. 127-36.

[6] Nakajima, S., et al., *3D MRI reconstruction for surgical planning and guidance*, in *Advanced Navigation in Neurosurgery*, E. Alexander and R.J. Maciunas, Editors. 1999, New York: Thieme. p. 137-145.

[7] Jones, M.W., *Facial reconstruction using volumetric data*, in *Proceedings of the Vision Modelling and Visualization Conference 2001*. 2001, Berlin: AKA GmbH. p. 135-150.

[8] Evison, M.P., *Computerised 3D facial reconstruction.* assemblage - the Sheffield graduate journal of archaeology, 1996. **1**(1).

[9] *Medical data storage and processing on the GRID*. 2002 [Accessed 2008 March 13]; Available from: http://creatis-www.insa-lyon.fr/MEDIGRID.

[10] Benkner, S., et al., *GEMSS: Grid-infrastructure for Medical Service Provision.* Methods of Information in Medicine, 2005. **44**(2): p. 177-181.

[11] Purdam, K., et al., *Grid computing and disclosure control.* New Review of Information Networking, 2004. **10**: p. 161-176.

[12] Banatre, J.-P., et al., *Future for European Grids: grids and service oriented knowledge utilities. Vision and research directions 2010 and beyond.*, D.D. Roure, Editor. 2006, Next Generation GRIDs Expert Group, Information Society Technologies (IST), European Commission, Office for Official Publications of the European Communities, Luxembourg. p. 60.

[13] Tomita, Y., et al., *Artificial neural network approach for selection of susceptible single nucleotide polymorphisms and construction of prediction model on childhood allergic asthma.* BMC Bioinformatics, 2004. **5**(1): p. 120.

[14] Bao, L. and Y. Cui, *Prediction of the phenotypic effects of nonsynonymous single nucleotide polymorphisms using structural and evolutionary information.* Bioinformatics, 2005. **10**(21): p. 2185-2190.

[15] Glaser, J., et al., *Advancing personalized health care through health information technology: an update from the American Health Information Community's Personalized Health Care Workgroup.* Journal of the American Medical Informatics Association, 2008. **15**(4): p. 391-396.

[16] Lin, Z., A.B. Owen, and R.B. Altman, *Fundamental tensions between genomic research and human subject privacy. Technical Report SMI-2003-1009*, in *Stanford Medical Informatics Reports*. 2003, Stanford University School of Medicine: Stanford. p. 21.

[17] Franke, A., et al., *GENOMIZER: an integrated analysis system for genome-wide association data.* Human Mutation, 2006. **27**(6): p. 583-588.

[18] Andrade, J., et al., *The use of grid computing to drive data-intensive genetic research.* European Journal of Human Genetics, 2007. **15**(6): p. 694-702.

[19] Lin, Z., A.B. Owen, and R.B. Altman, *Genomic research and human subject privacy.* Science, 2004. **305**(5681): p. 183.

[20] Turakulov, R. and S. Easteal, *Number of SNPS loci needed to detect population structure.* Hum Hered, 2003. **55**(1): p. 37-45.

[21] Ferraiolo, D.F., D.R. Kuhn, and R. Chandramouli, *Role-based access control*. Artech House computer security series. 2003, Boston, Mass., U.S.A.: Artech House. XVII, 316 S.

[22] Sandhu, R.S., et al., *Role-based access control models.* IEEE Computer 1996. **29**(2): p. 38-47.

[23] Sandhu, R. and J. Park, *Usage Control: a vision for next generation access control* in *Computer network security: Proceedings of the second international workshop on mathematical methods, models, and architectures for computer network security, MMM-ACNS 2003*, V.I. Gorodetski, L.J. Popyack, and V.A. Skormin, Editors. 2003, Berlin; New York: Springer. p. 17-31.

[24] Ferraiolo, D.F. and D.R. Kuhn. *Role based access control*. in *Proceedings of the 15th NIST-NSA National (U.S.A.) Computer Security Conference*. 1992.

[25] Park, J. and R. Sandhu, *The UCONABC usage control model.* ACM Transactions on Information and System Security (TISSEC), 2004. **7**(1): p. 128-174.

[26] Sandhu, R. and P. Samarati, *Access control: principles and practice.* IEEE Communications, 1994. **32**(9): p. 40-48.

[27] Sandhu, R., D. Ferraiolo, and R. Kuhn, *The NIST model for role-based access control: towards a unified standard*, in *Proceedings of the fifth ACM workshop on Role-based access control*. 2000, New York: ACM. p. 47-63.

[28] Duncan, G.T., S.A. Keller-McNulty, and S.L. Stokes, *Disclosure risk vs. data utility: the R-U confidentiality map. Technical Report Number 121*. 2001, National Institute of Statistical Sciences, NC, U.S.A.

[29] Willenborg, L.C.R.J. and T.d. Waal, *Elements of statistical disclosure control*. 2001, New York: Springer. xv, 261 p.

[30] Seitz, L., J.-M. Pierson, and L. Brunie, *Semantic access control for medical applications in grid environments*, in *Proceedings of the 9th International*

Euro-Par 2003 Parallel Processing Conference. 2003, Berlin; NewYork: Springer. p. 374-383.

[31] Roure, D.D., N.R. Jennings, and N.R. Shadbolt, *The semantic grid: past, present and future.* Proceedings of the IEEE, 2005 **93**(3): p. 669 - 681.

[32] Foster, I. and C. Kesselman, *The grid: blueprint for a new computing infrastructure*. 2nd ed. The Elsevier series in grid computing. 2004, Amsterdam; Boston: Morgan Kaufmann. xxvii, 748 p.

[33] Lim, H.B., et al., *Sensor grid: integration of wireless sensor networks and the grid.* Proceedings of the IEEE Conference on Local Computer Networks. 30th Anniversary, 2005: p. 91-99.

[34] Granzer, T., et al., *Providing remote access to robotic telescopes by adopting grid technology*, in *Proceedings of the German e-Science Conference*. 2007, Max-Planck-Gesellschaft eDoc Server.

[35] Nemeth, Z. and V. Sunderam, *Characterizing grids: attributes, definitions, and formalisms.* Journal of Grid Computing, 2003. **1**: p. 9-23.

[36] European Data Grid Consortium. *The datagrid project*. 2001 [Accessed 2008 June 20]; Available from: http://eu-datagrid.web.cern.ch/.

[37] Stanke, M., A. Tzvetkova, and B. Morgenstern, *AUGUSTUS at EGASP: using EST, protein and genomic alignments for improved gene prediction in the human genome.* Genome Biology, 2006. **7**(Supplement 1): p. S11.1-S11.8.

[38] *Globus Toolkit (GT)*. [Accessed 2008 May 26]; Available from: www.globus.org/.

[39] Oster, S., et al., *caGrid 1.0: An Enterprise Grid Infrastructure for Biomedical Research.* Journal of the American Medical Informatics Association, 2007: p. M2522.

[40] Berry, D., et al. *The Open Grid Services Architecture*. 2005 [Accessed 2008 February 20]; Available from: www.gridforum.org/documents/GWD-I-E/GFD-I.030.pdf.

[41] Bundesministerium für Bildung und Forschung (BMBF). *3. D-Grid call: BMBF-Förderbekanntmachung "Grid-Dienste für Wirtschaft und Wissenschaft"*. 2008 [Accessed 2008 May 26]; Available from: www.bmbf.de/foerderungen/12378.php.

[42] *InGrid*. 2005 [Accessed 2008 May 26]; Available from: www.ingrid-info.de.

[43] *Second D-Grid Security Workshop*. March 27-28, 2007. Goettingen.

[44] *Third D-Grid Security Workshop*. April 1-2, 2008. Goettingen.

[45] Sandhu, R., *Engineering authority and trust in cyberspace: the OM-AM and RBAC way*, in *Proceedings of the fifth ACM workshop on Role-based access control*. 2000, New York: ACM Press.

[46] Yao, W., *Trust management for widely distributed systems. Technical Report Number 608*. 2004, University of Cambridge: Cambridge. p. 191.

[47] Jiang, X., J.I. Hong, and J.A. Landay, *Approximate information flows: socially-based modeling of privacy in ubiquitous computing*, in *Proceedings*

of the 4th International Conference on Ubiquitous Computing. 2002, Berlin; New York: Springer. p. 176-193.

[48] Schumacher, M. and U. Roedig. *Security engineering with patterns*. in *Proceedings of the 8th Conference on Pattern Languages of Programs (PLoP 2001)*. 2001: Urbana; Champaign, Illinois, U.S.A.: Department of Computer Science at the University of Illinois at Urbana-Champaign

[49] Schumacher, M., *Security Engineering with Patterns: Origins, Theoretical Models, and New Applications*. 2003: New York: Springer. 208 p.

[50] Hu, V., R. Kuhn, and T. Xie, *Property verification for access control models via model checking. Technical Report TR-2008-1* 2008, North Carolina State University, Department of Computer Science: Raleigh, NC, U.S.A.

[51] Nagaratnam, N., et al. *The security architecture for open grid services*. 2002 [Accessed 2008 August 28]; Available from: www.cs.virginia.edu/%7Ehumphrey/ogsa-sec-wg/OGSA-SecArch-v1-07192002.pdf.

[52] Sloot, P., *Advances in grid computing--EGC 2005: European Grid Conference, Amsterdam, the Netherlands, February 14-16, 2005: revised selected papers*. Lecture notes in computer science. Vol. 3470. 2005: Berlin; New York: Springer. xxi, 1197 p.

[53] *Open Grid Services Architecture - Data Access and Integration Services (OGSA-DAI)*. [Accessed 2008 August 28]; Available from: www.ogsadai.org.uk/.

[54] Antonioletti, M., et al., *The design and implementation of Grid database services in OGSA-DAI.* Concurrency and Computation: Practice & Experience, 2005. **17**(2-4): p. 357-376.

[55] Foster, I., et al. *The physiology of the grid - An open grid services architecture for distributed systems integration*. [Accessed 2008 February 20]; Available from: www.globus.org/alliance/publications/papers/ogsa.pdf.

[56] *Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications).* Official Journal of the European Communities, Luxembourg, 2002(No L. 201): p. 37-47.

[57] Herveg, J.A.M. and Y. Poullet, *Directive 95/46 and the use of GRID technologies in the heathcare sector: selected legal issues*, in *Proceeding of the 1st Healthgrid conference*, S. Norager, Editor. 2003. p. 233-240.

[58] *Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data.* Official Journal of the European Communities, Luxembourg, 1995(No L. 281): p. 31-50.

[59] Tracy, C.S., G.C. Dantas, and R.E. Upshur, *Feasibility of a patient decision aid regarding disclosure of personal health information: qualitative evaluation of the Health Care Information Directive.* BMC Medical Informatics and Decision Making, 2004. **4**(1): p. 13.

[60] Herveg, J., *Does HealthGrid present specific risks with regard to data protection?*, in *From genes to personalized healthcare: grid solutions for the*

life sciences, N. Jacq, et al., Editors. 2007, Amsterdam; Washington, DC: IOS Press. p. 219-228.

[61] Herveg, J., *The ban on processing medical data in European law: consent and alternative solutions to legitimate processing of medical data in HealthGrid*, in *Challenges and opportunities of HealthGrids: Proceedings of Healthgrid 2006*, V. Hernández, et al., Editors. 2006, Amsterdam; Washington, DC: IOS Press. p. 107-116.

[62] Sax, U., Mohammed, Y., Viezens, F., Rienhoff, O., *Grid-Computing in der Biomedizinischen Forschung – Datenschutz und Datensicherheit*. Medizinische Informatik, Biometrie und Epidemiologie. Vol. 90. 2006, München: Medizin und Wissen Publ.Comp.

[63] Sax, U. and Y. Mohammed, *Data-related Challenges of Genotype - Phenotype studies*, in *Proceedings of the 51 Jahrestagung der GMDS* M. Löffler and A. Winter, Editors. 2006, Leipzig, Germany: Jutte Messedruck p. 100-102.

[64] Mohammed, Y., et al., *Rechtliche Aspekte bei Grid-Computing in der Medizin*, in *Rechtliche Aspekte der Telemedizin*, W. Niederlag, et al., Editors. 2006, Dresden, Germany: Health Academy. p. 235-245.

[65] Mohammed, Y., et al., *Shortcomings of current grid middlewares regarding privacy in HealthGrids*, in *From genes to personalized healthcare: grid solutions for the life sciences*, N. Jacq, et al., Editors. 2007, Amsterdam; Washington, DC.: IOS Press. p. 322-329.

[66] Drepper, J., Semler, S.C., Mohammed, Y., Sax, U., *Aktuelle Themen des Datenschutzes und der Datensicherheit in der biomedizinischen Forschung*, in *Grid-Computing in der Biomedizinischen Forschung – Datenschutz und Datensicherheit*, U. Sax, et al., Editors. 2006, Urban & Vogel: München. p. 25-36.

[67] Legré, Y., Mohammed, Y., Viezens, F., Rienhoff, O., Sax, U., *HealthGRID/ SHARE: grids for health - international interoperability*, in *Proceedings of the eHealth Conference 2007 - Experts' Special Interest Sessions*. 2008, Bonn, Germany: nanos Verlag. p. 81-99.

[68] Chakrabarti, A., *Grid computing security*. 2007, Berlin, New York: Springer. xiv, 331 p.

[69] Lorch, M., et al., *Conceptual grid authorization framework and classification*, M. Lorch, Editor. 2004, Global Grid Forum. p. 36.

[70] Vollbrecht, J., et al., *RFC 2904 - AAA Authorization framework*. 2000, The Internet Engineering Task Force (IETF). p. 35.

[71] Lampson, B.W., *Protection.* 5th Princeton Symposium on Information Science and Systems, Princeton University, March 1971, p. 437-443. Reprinted in ACM Operating Systems Review, 1971. **8**(1): p. 18-24.

[72] Saltzer, J.H. and M.D. Schroeder, *The protection of information in computer systems.* Proceedings of the IEEE, 1975. **63**(9): p. 1278-1308.

[73] Needham, R.M. and R.D.H. Walker, *The Cambridge CAP computer and its protection system*, in *Proceedings of the 6th ACM Symposium on Operating Systems Principles*. 1977, New York: ACM Press. p. 1-10.

[74] Bell, D.E. and L.J. La Padula, *Secure computer systems: mathematical foundations and model. Technical Report M74-244*. 1974, MITRE Corporation: Bedford, Mass., U.S.A.

[75] Biba, K.J., *Integrity considerations for secure computer systems. Technical Report MTR-3153*. 1977, MITRE Corporation: Bedford, Mass., U.S.A. p. 69.

[76] Clark, D.D. and D.R. Wilson, *A comparison of commercial and military computer security policies*, in *Proceedings of the 1987 IEEE Symposium on Security and Privacy*. 1987, Washington, DC, U.S.A.: IEEE Computer Society Press. p. 184-194.

[77] Brewer, D.F.C. and M.J. Nash, *The Chinese Wall security policy*, in *Proceedings of 1989 IEEE Symposium on Security and Privacy*. 1989, Washington, DC, U.S.A.: IEEE Computer Society Press. p. 206-214.

[78] Krutz, R.L. and R.D. Vines, *The CISSP prep guide*. Gold ed. 2003: Indianapolis, Ind., U.S.A.: Wiley. xxviii, 944 p.

[79] United States. Department of Defense, *Trusted computer system evaluation criteria*. Dtd 15 Aug. 83 ed. 1985, Washinton, DC: Department of Defense. V, 121 S.

[80] Ferraiolo, D.F., D.M. Gilbert, and N. Lynch, *An examination of federal and commercial access control policy needs*, in *Proceedings of the 16th NIST-NCSC National Computer Security Conference*. 1993, National Institute of Standards and Technology - National Computer Security Center. p. 107-116.

[81] Ferraiolo, D., R. Kuhn, and R. Sandhu, *RBAC standard rationale: comments on "A Critique of the ANSI Standard on Role-Based Access Control"* IEEE Security & Privacy 2007. **5**(6): p. 51-53.

[82] McPherson, D. and Microsoft Corporation. *Role-based access control for multi-tier applications using authorization manager - White paper*. July 31 2004 [Accessed 2008 June 11]; Available from: http://technet2.microsoft.com/WindowsServer/en/library/.

[83] Thomas, R.K. and R.S. Sandhu, *Task-Based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-Oriented Autorization Management*, in *Proceedings of the IFIP TC11 WG11.3 Eleventh International Conference on Database Securty XI: Status and Prospects*. 1998, London, U.K.: Chapman & Hall.

[84] Sandhu, R.S., *Lattice-Based Access Control Models.* Computer, 1993. **26**(11): p. 9-19.

[85] Sandhu, R. and Q. Munawer, *How to do discretionary access control using roles*, in *Proceedings of the third ACM workshop on Role-based access control*. 1998, New York, U.S.A.: ACM Press.

[86] Osborn, S., R. Sandhu, and Q. Munawer, *Configuring role-based access control to enforce mandatory and discretionary access control policies.* ACM Transactions on Information and System Security, 2000. **3**(2): p. 85-106.

[87] *RBAC Standards*. National Institute of Standards and Technology (NIST) 2004 [Accessed 2008 August 28]; Available from: http://csrc.nist.gov/groups/SNS/rbac/standards.html.

[88] Li, N., J.-W. Byun, and E. Bertino, *A critique of the ANSI standard on role-based access control.* IEEE Security & Privacy 2007 **5**(6): p. 41-49.

[89] The National Information Assurance Partnership (NIAP) Common Criteria Evaluation and Validation Scheme for IT Security (CCEVS). *Validated Product - Red Hat Enterprise Linux Version 5* 2007 [Accessed 2008 June 20]; Available from: www.niap-ccevs.org/cc-scheme/st/?vid=10165.

[90] Sun Microsystems Inc., *RBAC in the Solaris Operating Environment - White Paper*. 2000, Palo Alto, CA, U.S.A.: Sun Microsystems p. 39.

[91] Mossakowski, T., M. Drouineaud, and K. Sohr, *A temporal-logic extension of role-based access control covering dynamic separation of duties*, in *Proceedings of the 10th International Symposium on Temporal Representation and Reasoning and 4th International Conference on Temporal Logic*. 2003, Washington, DC, U.S.A.: IEEE Computer Society Press. p. 83-90.

[92] Abadi, M., et al., *A calculus for access control in distributed systems.* ACM Transactions on Programming Languages and Systems 1993. **15**(4): p. 706-734.

[93] Needham, R.M. and M.D. Schroeder, *Using encryption for authentication in large networks of computers.* Communication ACM, 1978. **21**(12): p. 993-999.

[94] Miller, S.P., et al., *Kerberos authentication and authorization system: Project Athena Technical Plan, section E.2.1*. 1988, Cambridge, Mass., U.S.A.: Massachusetts Institute of Technology. p. 36.

[95] Wulf, W., et al., *HYDRA: the kernel of a multiprocessor operating system.* Communication ACM, 1974. **17**(6): p. 337-345.

[96] Needham, R.M. and A.J. Herbert, *The Cambridge distributed computing system*. International Computer Science Series. 1982, London; Reading, Mass., U.S.A.: Addison-Wesley x, 170 p.

[97] Tanenbaum, A.S., et al., *Experiences with the Amoeba distributed operating system.* Communication ACM, 1990. **33**(12): p. 46-63.

[98] Gong, L., *On security in capability-based systems.* ACM SIGOPS (Special Interest Group on Operating Systems) Operating Systems Review, 1989. **23**(2): p. 56-60.

[99] Blaze, M., J. Feigenbaum, and L. Jack. *Decentralized trust management*. in *Proceedings of the 17th IEEE Symposium on Security and Privacy*. 1996: Washington, DC, U.S.A.: IEEE Computer Society Press.

[100] Herzberg, A., et al., *Access control meets public key infrastructure, or: assigning roles to strangers*, in *Proceedings of the 2000 IEEE Symposium on Security and Privacy*. 2000, Washington, DC, U.S.A.: IEEE Computer Society Press. p. 2-14.

[101] Winsborough, W.H., K.E. Seamons, and V.E. Jones, *Automated trust negotiation*, in *Proceedings of DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00*. 2000, Los Alamitos, CA, U.S.A.: IEEE Computer Society Press. p. 88-102.

[102] Gong, L., *A secure identity-based capability system*, in *Proceedings of the 1989 IEEE Symposium on Security and Privacy*. 1989, Washington, DC, U.S.A.: IEEE Computer Society Press. p. 56-63.

[103] Sibert, O., D. Bernstein, and D.V. Wie, *DigiBox: a self-protecting container for information commerce*, in *Proceedings of the 1st conference on USENIX Workshop on Electronic Commerce - Volume 1*. 1995, Berkeley, CA, U.S.A.: USENIX Association. p. 171-183.

[104] Wang, X., et al., *XrML - eXtensible rights Markup Language*, in *Proceedings of the 2002 ACM workshop on XML security*. 2002, New York, NY, U.S.A.: ACM.

[105] Feigenbaum, J., *Digital rights management: ACM CCS-9 Workshop DRM 2002, Washington, DC, U.S.A., November 18, 2002: revised papers*. Lecture Notes in Computer Science. Vol. 2696. 2003, Berlin; New York: Springer. x, 220 p.

[106] LaMacchia, B.A., *Key challenges in DRM: an industry perspective*, in *Digital rights management: ACM CCS-9 workshop DRM 2002: revised papers*, J. Feigenbaum, Editor. 2003, Berlin; New York: Springer. p. 51-60.

[107] Keen, R., *Broadcasting medical image objects with digital rights management (patent)*. 2007, Assignee 20070270695: U.S.A.

[108] Koster, R. and W. Jonker, *Digital rights management for retrieving medical data from a server (patent)*, World Intellectual Property Organization. 2007, WO/2007/105148: Netherlands.

[109] Duncan, C., et al., *Digital rights management*, in *JISC DRM Study (study carried out by Intrallect Ltd. on behalf of JISC)*. 2004, Intrallect Ltd.: Linlithgow, U.K.

[110] Pearlman, L., et al., *A community authorization service for group collaboration*, in *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*. 2002, Washington, DC, U.S.A.: IEEE Computer Society Press. p. 50-59.

[111] The Globus Security Team, *Globus toolkit version 4 Grid Security Infrastructure: a standards perspective*, V. Welch, Editor. 2005, Urbana; Champaign, Illinois, U.S.A.: National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign. p. 8.

[112] Welch, V., et al., *X.509 proxy certificates for dynamic delegation*, in *Proceedings of The 3rd Annual PKI R&D Workshop*. 2004, National Institute of Standards and Technology Virtual Library.

[113] Neuman, B.C. and T. Tsapos, *Kerberos: an authentication service for computer networks.* IEEE Communications, 1994. **32**(9): p. 33-38.

[114] *GT 4.0 Security: Key Concepts*. 2008 [Accessed 2008 March 11]; Available from: www.globus.org/toolkit/docs/4.0/security/key-index.html

[115] Tuecke, S., et al. *(RFC3820) Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile*. Request for Comments (RFC) 2004 [Accessed 2008 March 2008]; Available from: http://www.faqs.org/rfcs/rfc3820.html.

[116] *GT 4.0 Security*. 2008 [Accessed 2008 March 11]; Available from: www.globus.org/toolkit/docs/4.0/security/.

[117] *Proxy Certificates Types*. 2007 [Accessed 2008 March 14]; Available from: http://dev.globus.org/wiki/Security/ProxyCertTypes.

[118] *Realizing the information future: the Internet and beyond*. 1994, Washington, D.C., U.S.A.: National Academy Press. xv, 301 p.

[119] *GT 4.0 Security: Delegation Service*. [Accessed 2008 June 30]; Available from: www.globus.org/toolkit/docs/4.0/security/.

[120] Alfieri, R., et al., *VOMS, an authorization system for virtual organizations*, in *Grid computing: first European Across Grids Conference, Santiago de Compostela, Spain, February 13-14, 2003 : revised papers. Lecture notes in computer science, 2970*, F. Fernández Rivera, Editor. 2004, Berlin ; New York: Springer. p. 33-40.

[121] Thompson, M.R., A. Essiari, and S. Mudumbai, *Certificate-based authorization policy in a PKI environment.* ACM Transactions on Information and System Security, 2003. **6**(4): p. 566-588.

[122] Chadwick, D.W. and A. Otenko, *The PERMIS X.509 role based privilege management infrastructure*, in *Proceedings of the seventh ACM symposium on access control models and technologies*. 2002, Monterey, California, U.S.A.: ACM Press. p. 135-140.

[123] Seitz, L., *Design and Implementation of Secure Mechanisms for Sharing Confidential Data; Application to the Management of Biomedical Data in a Grid Computing Environment*. Informatique et Information pour la Société, L'Institut National des Sciences Appliquées de Lyon. PhD. Thesis advisor: B. Elisa, et al. January 11 2005. 201 p.

[124] Jin, H., et al., *RB-GACA: an RBAC based grid access control architecture.* International Journal of Grid and Utility Computing, 2005. **1**: p. 61-70.

[125] Kane, K. and J.C. Browne, *On classifying access control implementations for distributed systems*, in *Proceedings of the eleventh ACM symposium on access control models and technologies*, D. Ferraiolo and I. Ray, Editors. 2006, New York: ACM Press. p. 29-38

[126] Pearlman, L., et al. *The community authorization service: Status and future*. in *Proceedings of the Conference for Computing in High Energy and Nuclear Physics*. 2003: Stanford, CA, U.S.A.: Stanford University.

[127] Chadwick, D. and O. Otenko, *A comparison of the Akenti and PERMIS authorization infrastructures in ensuring security in IT infrastructures*, in *Proceedings of the ITI First International Conference on Information and Communications Technology (ICICT 2003)* M.T. El-Hadidi, Editor. 2003, Cairo, Egypt: Cairo University. p. 5-26.

[128] Health Level Seven, I. *HL7, Inc.* 1987 [Accessed 2008 January 10]; Available from: www.hl7.org.

[129] *Health Level Seven Australia*. [Accessed 2008 January 10]; Available from: www.hl7.com.au/FAQ.htm#HL7-Inc.

[130] Science Applications International Corporation (SAIC), *Healthcare role-based access control task force charter*. 2003, La Jolla, CA, U.S.A.: Science Applications International Corporation (SAIC). p. 8.

[131] HL7 Security Technical Committee, *Role Based Access Control (RBAC) healthcare permission catalog*. 2007, HL7. p. 31.

[132] HL7 Security Technical Committee, *HL7 Role-based Access Control (RBAC) Role Engineering Process*. November 2005, HL7.

[133] Blobel, B., et al., *Modelling privilege management and access control.* International Journal of Medical Informatics, 2006. **75**(8): p. 597-623.

[134] Kapsalisa, V., et al., *A dynamic context-aware access control architecture for e-services.* Computers & Security, 2006. **25**(7): p. 507-521

[135] Hu, J. and A.C. Weaver, *Dynamic context-aware access control for distributed healthcare applications*, in *First Workshop Pervasive Security, Privacy and Trust (PSPT '04).* 2004. p. 8.

[136] Ni, Q., et al., *Privacy-aware role based access control*, in *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies*. 2007, New York: ACM Press. p. 41-50.

[137] Chakraborty, S. and I. Ray, *TrustBAC: integrating trust relationships into the RBAC model for access control in open systems*, in *Proceedings of the eleventh ACM symposium on access control models and technologies*. 2006, New York, NY, U.S.A.: ACM Press. p. 49-58.

[138] Black, P.E. *Dictionary of algorithms and data structures*. 2007 [Accessed 2008 June 12]; Available from: www.nist.gov/dads/.

[139] Galton, A., *Temporal Logic*, in *Stanford Encyclopedia of Philosophy* E.N. Zalta, Editor. 2008, Stanford, CA, U.S.A.: The Metaphysics Research Lab, Center for the Study of Language and Information, Stanford University.

[140] Pnueli, A., *The temporal logic of programs*, in *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science*. 1977, Washington, DC, U.S.A.: IEEE Computer Society Press. p. 46-67.

[141] Voronkov, A. *Lecture notes in Logic in Computer Science*. 2008 [Accessed 2008 July 20]; Available from: www.voronkov.com.

[142] Clarke, E.M., O. Grumberg, and D. Peled, *Model checking*. 1999, Cambridge, Mass.: MIT Press. xiv, 314 p.

[143] Siewe, F., A. Cau, and H. Zedan, *A compositional framework for access control policies enforcement*, in *Proceedings of the 2003 ACM workshop on Formal methods in security engineering*. 2003, New York, NY, U.S.A.: ACM Press. p. 32-42.

[144] Zhang, X., et al., *A logical specification for usage control*, in *Proceedings of the ninth ACM symposium on Access control models and technologies*. 2004, New York, NY, U.S.A.: ACM Press. p. 1-10.

[145] Zhang, X., *Formal Model And Analysis Of Usage Control*. Department of Information and Software Engineering, George Mason University. Doctor of Philosophy in Information Technology. Thesis advisor: R. Sandhu. Summer Semester 2006. 176 p.

[146] Park, J. and R. Sandhu, *Towards usage control models: beyond traditional access control*, in *Proceedings of the seventh ACM symposium on access control models and technologies*. 2002, New York: ACM Press. p. 57-64.

[147] Mohammed, Y., *A question about the UCON model*. November 7, 2007, Email communication with R. Sandhu and J. Park Goettingen.

[148] Lamport, L., *The temporal logic of actions.* ACM Transactions on Programming Languages and Systems (TOPLAS) 1994. **16**(3): p. 872-923.

[149] Cohen, E. and S. Narain, *Temporal Logic*. Wiley Encyclopedia of Electrical and Electronics Engineering. Vol. 24. 1999: Wiley - Interscience. 17616 p.

[150] Bozeman, B., *Technology transfer and public policy: a review of research and theory.* Research Policy, 2000. **29**(4-5): p. 627-655.

[151] Reisman, A., *Technology transfer: A taxonomic view.* The Journal of Technology Transfer, 1989. **14**(3): p. 31-36.

[152] Reisman, A., *Transfer of technologies: a cross-disciplinary taxonomy.* Omega, 2005. **33**(3): p. 189-202.

[153] *D-Grid initiative* 2005 [Accessed 2007 March 14]; Available from: www.d-grid.de.

[154] Neuroth, H., M. Kerzel, and W. Gentzsch, *German Grid Initiative D-Grid*. 2007, Göttingen, Germany: University Verlag Göttingen, Niedersächsische Staats- und Universitätsbibliothek.

[155] *Information Society Germany 2006*. 2003, Federal Ministry of Economics and Labour and Federal Ministry of Education and Research. p. 92.

[156] *MediGRID* 2005 [Accessed 2008 March 14]; Available from: www.medigrid.de.

[157] *First D-Grid Security Workshop*. March 21-22, 2006. Goettingen.

[158] Mohammed, Y., et al., *The integration of grid resources into a portal for research collaboratories*, in *Proceedings of the Medinfo 2007 conference*. 2007, Amsterdam; Washington, DC.: IOS Press. p. 335-339.

[159] Rienhoff, O., *A legal framework for security in European health care telematics*. Studies in health technology and informatics. Vol. 74. 2000, Amsterdam; Washington, DC: IOS Press. x, 192 p.

[160] Jajodia, S., M. Kudo, and V.S. Subrahmanian, *Provisional authorization*, in *Advances in information security; E-commerce security and privacy*, A.K. Ghosh, Editor. 2001, Boston: Kluwer Academic Publishers. p. xxi, 163 p.

[161] Blobel, B. and P. Pharow, *Results of European projects improving security of distributed health information systems*, in *Proceedings of MEDINFO 98*, B. Cesnik, A. McCray, and J.-R. Scherrer, Editors. 1998, Amsterdam, Berlin, Oxford, Tokyo, Washington DC: IOS Press p. 1119-1123

[162] Soltwisch, R. and D. Hogrefe, *A survey on network security. Technical Report Number IFI-TB-2004–04*. 2004, Göttingen, Germany: Georg-August-Universität Göttingen - Institut für Informatik. p. 181.

[163] Barka, E. and R. Sandhu, *A role-based delegation model and some extensions*, in *Proceedings of 23rd National Information Systems Security Conference*. 2000, National Institute of Standards and Technologie and National Computer Security Center. p. 101-111.

[164] Keahey, K. and V. Welch, *Fine-grain authorization for resource management in the grid environment*, in *Proceedings of Grid Computing--GRID 2002: 3rd International Workshop*. 2002, Berlin; New York: Springer. p. 199-206

[165] Ackoff, R.L., *From Data to Wisdom.* Journal of Applied Systems Analysis, 1989. **16**: p. 3-9.

[166] Foster, I., N.R. Jennings, and C. Kesselman, *Brain meets brawn: why grid and agents need each other*, in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*. 2004, Washington, DC, U.S.A.: IEEE Computer Society Press. p. 8-15.

[167] *IVOM Workshop (Interoperability and Integration of VO-Management Technologies in D-Grid)*. February 19, 2008. Hannover.

[168] *GT 4.0 Data Management* 2008 [Accessed 2008 March 12]; Available from: www.globus.org/toolkit/docs/4.0/data/.

[169] *Web Services Resource Framework (WSRF)*. 2006 [Accessed 2008 March 12]; Available from: www.globus.org/wsrf/.

[170] *Game Theory*. 2008 [Accessed 2008 June 24]; Available from: http://encarta.msn.com.

[171] Harrald, L., *Artificial life: model for musical innovation*, in *Proceedings of the Australian Conference on Artificial Life (ACAL2003)*. 2003, Canberra, Australia: University of New South Wales. p. 128-141.

[172] Lang, B., et al., *Attribute based access control for grid computing.* Preprint ANL/MCS-P1367-0806, 2006.

[173] Franklin, S. and A. Graesser, *Is it an agent, or just a program?: a taxonomy for autonomous agents*, in *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*. 1997, Berlin: Springer. p. 21-35.

[174] Brooks, R.A., *Intelligence without reason*. 1991, Cambridge, MA, U.S.A.:Massachusetts Institute of Technology.

[175] Wooldridge, M. and N. Jennings, *Intelligent Agents: Theory and Practice.* Knowledge Engineering Review 1995. **10**(2): p. 115-152.

[176] Shoham, Y., *An overview of agent-oriented programming*, in *Software agents*. 1997, Cambridge, MA, U.S.A.: MIT Press. p. 271-290.

[177] Finin, T., et al., *KQML as an agent communication language*, in *Proceedings of the third international conference on information and knowledge management*. 1994, New York, NY, U.S.A.: ACM Press. p. 456-463.

[178] Flores-Mendez, R.A., *Towards a standardization of multi-agent system framework.* Crossroads, 1999. **5**(4): p. 18-24.

[179] Wooldridge, M.J., *An introduction to multiagent systems*. 2002: New York: John Wiley & Sons. xviii, 348 p.

[180] Russell, S.J. and P. Norvig, *Artificial intelligence: a modern approach*. Prentice Hall series in artificial intelligence. 1995, Englewood Cliffs, N.J.: Prentice Hall. xxviii, 932 p.

[181] Hewitt, C., *Offices are open systems.* ACM transactions on Information Systems, 1986. **4**(3): p. 271-287.

[182] Sycara, K., *Multiagent Systems.* AI Magazine, 1998. **10**(2): p. 79-93.

[183] Vidal, J.M. *Fundamentals of Multiagent Systems*. 2007 [Accessed 2008 July 6]; Available from: www.multiagent.com/fmas.

[184] Alur, R., T.A. Henzinger, and O. Kupferman, *Alternating-time temporal logic.* Journal of the ACM, 2002. **49**(5): p. 672-713.

[185] Halpern, J.Y. and R. Fagin, *Modelling knowledge and action in distributed systems*, in *Proceedings of the International Conference on Concurrency*. 1988, Berlin; New York: Springer. p. 18-32.

[186] Bing, J., et al., *Report on legal issues. Deliverable 15*. 2005, TrustCoM Consortium. p. 38.

[187] Foster, I., C. Kesselman, and S. Tuecke, *The anatomy of the grid: enabling scalable virtual organizations.* International Journal of High Performance Computing Applications, 2001. **15**(3): p. 200-222.

[188] Mazzeschi, M., *The virtual organisation*, in *Proceedings of the 7th International Conference on Concurrent Enterprising*. 2001, Nottingham, U.K.: University of Nottingham. p. 331-336.

[189] Weitzenboeck, E.M., *Building a legal framework for a virtual organisation in the maritime domain: the MARVIN experience*, in *Proceedings of the 7th International Conference on Concurrent Enterprising*. 2001, Nottingham, U.K.: University of Nottingham. p. 337-346.

[190] Schoubroeck, C.V., et al., *A Legal Taxonomy on Virtual Enterprises*, in *Proceedings of the 7th International Conference on Concurrent Enterprising*. 2001, Nottingham, U.K.: University of Nottingham. p. 356-364.

[191] Arnold, O., et al., *Virtuelle Unternehmen als Unternehmenstyp der Zukunft.* Handbuch der modernen Datenverarbeitung - Theorie und Praxis der Wirtschaftsinformatik, 1995. **185**(32): p. 8-23.

[192] Berwanger, E., *The legal classification of virtual corporation according to german law.* Journal of Organizational Virtualness, 1999. **1**: p. 158-170.

[193] Berwanger, E., *Der Gesellschaftsvertrag eines virtuellen Unternehmens*. Westfälische WIlhelms-Universität zu Münster. PhD. Thesis advisor: Hoeren/Schlüter. 2000. 233 p.

[194] Goldberg, R.P., *Architecture of virtual machines*, in *Proceedings of the workshop on virtual computer systems*. 1973, New York, NY, U.S.A: ACM. p. 74-112.

[195] James, E.S. and N. Ravi, *The architecture of virtual machines.* Computer, 2005. **38**(5): p. 32-38.

[196] Foster, I., et al., *Virtual clusters for grid communities*, in *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)* 2006, Washington, DC, U.S.A.: IEEE Computer Society Press. p. 513-520.

[197] Keahey, K., et al., *Virtual workspaces for scientific applications.* Journal of Physics: Conference Series, 2007. **78**.

[198] Keahey, K., K. Doering, and I. Foster. *From sandbox to playground: dynamic virtual environments in the grid*. in *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*. 2004: Washington, DC, U.S.A: IEEE Computer Society Press.

[199] Ormandy, T., *An empirical study into the security exposure to hosts of hostile virtualized environments*, in *CanSecWest 2007*. 2007, Google Inc.

[200] Ferrie, P., *Attacks on virtual machines emulators*, in *Association of anti Virus Asia Researchers (AVAR) Conference*. 2006.

[201] Ferrie, P., *Attacks on more virtual machine emulators*, in *Symantec Advanced Threat Research*. 2007.

[202] Doyle, P., *Confidentiality, disclosure, and data access: theory and practical application for statistical agencies*. 1st ed. 2001, Amsterdam; New York: North-Holland. x, 452 p.

[203] Elliot, M., K. Purdam, and D. Smith, *Confidential data access using grid computing: an outline of the issues and possible solutions*, in *Proceedings of the First International Conference on e-Social Science*. 2005, Manchester, U.K.: National Centre for e-Social Science, University of Manchester.

[204] Rienhoff, O., *The SIREN legal workshops: list of urgent legal actions for telemedicine.* Studies in Health Technology and Informatics, 1999. **64**: p. 61-64.

[205] Allaert, F.A., *Security standards for healthcare information systems: a perspective from the EU ISIS MEDSEC Project*. Studies in health technology and informatics Vol. 69. 2002, Amsterdam; Oxford: IOS Press. 239 p.

[206] *Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures.* Official Journal of the European Communities, Luxembourg, 2000(No L. 13): p. 12–20.

[207] *Gesetz über Rahmenbedingungen für elektronische Signaturen*. 2001, Bundesministerium der Justiz, Bundesrepublik Deutschland: Berlin.

[208] Hes, R. and J.J. Borking, *Privacy-enhancing technologies: the path to anonymity*. Revised ed. Achtergrondstudies en verkenningen; 11. 1998, The Hague: Registratiekamer. 54 p.

[209] Federrath, H., *Privacy enhanced technologies: methods – markets – misuse* in *Trust, privacy and security in digital business* S. Katsikas, J. Lopez, and G. Pernul, Editors. 2005, Berlin; Heidelberg: Springer. p. 1-9.

[210] Borking, J.J. *Privacy Standards for Trust*. 27th International Conference on Privacy and Personal Data Protection 2005 September 03 [Accessed 2008 June 30´]; Available from: www.privacyconference2005.org/.

[211] Sax, U., *Modellierung ausgewählter Sicherheitsaspekte der "ambulant-stationären Verzahnung" im deutschen Gesundheitswesen*. Mathematisch-Naturwissenschaftliche Fakultäten, Georg-August-Universität Göttingen. PhD. Thesis advisor: O. Rienhoff and R. Schaback. 2002. 126 p.

[212] van der Haak, M., *Architekturkonzepte für einrichtungsübergreifende elektronische Patientenakten am Beispiel des Tumorzentrums Heidelberg/Mannheim*. Medizinische Biometrie und Informatik, Universitaetsklinkum Heidelberg. Dr. sc. hum. Thesis advisor: T. Wetter. June 27 2006. 231 p.

[213] Sax, U. and S. Schmidt, *Integration of genomic data in Electronic Health Records--opportunities and dilemmas.* Methods of Information in Medicine, 2005. **44**(4): p. 546-550.

[214] *Strafgesetzbuch*. 1998 Bundesministerium der Justiz, Bundesrepublik Deutschland: Berlin.

[215] Mand, E., *Datenschutz in Medizinnetzen.* MedR Medizinrecht, 2003. **21**(7): p. 393-400.

[216] Schütze, B., *Rechtliche Rahmenbedingungen der Teleradiologie.* Der Radiologe, 2007. **47**(2): p. 157-162.

[217] Welch, V., et al., *Security for Grid Services*, in *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03)*. 2003 Washington, DC, U.S.A.: IEEE Computer Society Press. p. 48-57.

[218] Rajasekar, A., et al., *Storage resource broker - managing distributed data in a grid.* Computer Society of India Journal, 2003. **33**(4): p. 42-54

[219] *Storage Resource Broker* 2006 [Accessed 2008 March 11]; Available from: www.sdsc.edu/srb/index.php/.

[220] Karasavvas, K., et al., *Introduction to OGSA-DAI services*, in *Scientific applications of grid computing*. 2005, Berlin; Heidelberg: Springer. p. 1-12.

[221] Herrero, P., M.S. Pérez, and V. Robles, *Scientific applications of grid computing: first international workshop, SAG 2004, revised selected and invited papers*. Lecture Notes in Computer Science. Vol. 3458. 2005, Berlin; New York: Springer. x, 208 p.

[222] Graham, S., et al. *Web Services Resource Framework (WSRF)*. 2006 [Accessed 2008 March 11]; Available from: www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf.

[223] *GT Execution Management: Grid Resource Allocation Management (GRAM)*. [Accessed 2008 March 12]; Available from: www.globus.org/toolkit/gram/.

[224] *GT Information Services: Monitoring & Discovery System (MDS)*. [Accessed 2008 March 11]; Available from: www.globus.org/toolkit/mds/.

[225] Krefting, D., et al., *MediGRID: Towards a user friendly secured grid infrastructure.* Future Generation Computer Systems, 2008(In Press).

[226] Stanke, M. and B. Morgenstern, *AUGUSTUS: a web server for gene prediction in eukaryotes that allows user-defined constraints.* Nucleic Acids Research, 2005. **33**(Web Server issue): p. W465-7.

[227] *MEDIGRID*. 2002-2005 [Accessed 2006 December 27]; Available from: www.creatis.insa-lyon.fr/MEDIGRID/.

[228] Montagnat, J., V. Breton, and I. Magnin, *Partitioning medical image databases for content-based queries on a grid.* Methods of Information in Medicine, 2005. **44**(2): p. 154-160.

[229] Seitz, L., et al., *Authentication and Authorisation Prototype on the µgrid for Medical Data Management*, in *Proceedings of the HealthGrid 2005 Conference - From Grid to Healthgrid* T. Solomonides, et al., Editors. 2005, Amsterdam; Washington, DC: IOS Press. p. 222 - 233.

[230] *µgrid - MEDIGRID*. 2005 [Accessed 2008 March 11]; Available from: www.creatis.insa-lyon.fr/MEDIGRID/.

[231] Seitz, L., J.M. Pierson, and L. Brunie, *Encrypted storage of medical data on a grid.* Methods of Information in Medicine, 2005. **44**(2): p. 198-201.

[232] *MammoGrid* 2005 [Accessed 2007 March 14]; Available from: www.mammogrid.com.

[233] Amendolia, S.R., et al., *Deployment of a grid-based medical imaging application*, in *From Grid to Healthgrid: Proceedings of Healthgrid 2005*. 2005, Amsterdam; Oxford: IOS Press. p. 59-69.

[234] Estrella, F., et al., *Experiences of engineering grid-based medical software.* International Journal of Medical Informatics, 2006. **76**(8): p. 621-632.

[235] Estrella, F., R. McClatchey, and D. Rogulin, *The MammoGrid virtual organisation - federating distributed mammograms.* Studies in Health Technology and Informatics, 2005. **116**: p. 935-940.

[236] *Grid-Enabled Medical Simulation Services (GEMSS)*. 2005 [Accessed 2008 March 13]; Available from: www.gemss.de.

[237] Middleton, S.E., et al., *GEMSS: Privacy and security for a medical grid.* Methods of Information in Medicine, 2005. **44**(2): p. 182-185.

[238] Lin, K., G. Daemer, and B.A. Hamilton, *caBIG security technology evaluation - White paper*. 2006, National Cancer Institute, Center for Bioinformatics. p. 114.

[239] HL7 Security Technical Committee, *Role Based Access Control (RBAC) role engineering overview* 2007, HL7. p. 2.

[240] Simpson, A., D. Power, and M. Slaymaker, *On tracker attacks in health grids*, in *Proceedings of the 2006 ACM symposium on Applied computing*. 2006, New York, NY, U.S.A.: ACM Press. p. 209-216.

[241] Espert, I.B., et al. *HealthGrid - White Paper* 2004 [Accessed 2008 June 05]; Available from: http://initiative.healthgrid.org/fileadmin/whitepaper/HealthGrid_whitepaper_full.pdf.

[242] Solomonides, T., *From grid to healthgrid: proceedings of Healthgrid 2005*. Studies in health technology and informatics. Vol. 112. 2005, Amsterdam; Washington, DC: IOS Press. xi, 324 p.

[243] Hernández, V. and I. Blanquer, *Challenges and opportunities of healthgrids: proceedings of HealthGrid 2006*. 2006, Amsterdam; Washington, DC: IOS Press. xii, 407 p.

[244] Jacq, N., et al., *From genes to personalized healthcare: grid solutions for the life sciences; Proceedings of HealthGrid 2007*. Studies in Health Technology and Informatics. Vol. 126. 2007, Amsterdam; Washingaton, DC: IOS Press. XIV, 341 S.

[245] Dolin, R.H., et al., *HL7 Clinical document architecture, Release 2.* Journal of the American Medical Informatics Association, 2006. **13**(1): p. 30-39.

[246] Zhao, Y., M. Wilde, and I. Foster, *Applying the virtual data provenance model* in *Provenance and annotation of data: International Provenance and Annotation Workshop; revised selectes papers. Lecture Notes in Computer Science. Vol 4145*, L. Moreau and I. Foster, Editors. 2006, Berlin; New York: Springer. p. 148-161.

[247] Moreau, L. and I. Foster, *Provenance and annotation of data: International Provenance and Annotation Workshop, IPAW 2006, revised selected papers*. Lecture Notes in Computer Science. Vol. 4145. 2006: Berlin; New York: Springer. xi, 288 p.

[248] Álvarez, S., et al., *Applying provenance in distributed organ transplant management*, in *Provenance and annotation of data: International Provenance and Annotation Workshop; revised selectes papers. Lecture Notes in Computer Science. Vol. 4145*, L. Moreau and I. Foster, Editors. 2006, Berlin; New York: Springer. p. 28-36.

[249] Chen, L., et al., *A proof of concept: Provenance in a service oriented architecture.*, in *Proceedings of the U.K. e-Science All Hands Meeting 2005*, S.J. Cox and D.W. Walker, Editors. 2005, Swindon, U.K.: Engineering and Physical Sciences Research Council (EPSRC). p. 247-281.

[250] *Provenance-Aware Service-Oriented Architecture*. 2004-2007 [Accessed 2008 January 04]; Available from: http://twiki.pasoa.ecs.soton.ac.uk/bin/view/PASOA/WebHome.

[251] *The Provenance Project: "Enabling and supporting provenance in grids for complex problems"*. 2004-2006 [Accessed 2008 January 04]; Available from: www.gridprovenance.org/.

[252] Kifor, T., et al., *Provenance in agent-mediated healthcare systems.* IEEE Intelligent Systems, 2006. **21**(6): p. 38-46

[253] *AG Datenschutz der TMF*. 2006 [Accessed 2006 March 16]; Available from: www.tmf-ev.de/site/DE/int/AG/DS/container_ag_ds.php.

[254] *Telematikplattform für Medzinische Forschungsnetze (TMF)*. 2006 [2006 December 28]; Available from: www.tmf-ev.de.

[255] Bergmann, J., et al., *An e-consent-based shared EHR system architecture for integrated healthcare networks.* International Journal of Medical Informatics, 2007. **76**(2-3): p. 130-136.

[256] de Riese, M., et al. *The dCache Book*. [Accessed 2008 August 28]; Available from: www.dcache.org/manuals/Book/Book-a4.pdf.

[257] *Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls.* Nature, 2007. **447**(7145): p. 661-678.

[258] Hirschhorn, J.N. and M.J. Daly, *Genome-wide association studies for common diseases and complex traits.* Nature Reviews Genetics, 2005. **6**(2): p. 95-108.

[259] van Halteren, A., et al., *MobiHealth: ambulant patient monitoring over next generation public wireless networks.* Studies in Health Technology and Informatics, 2004. **106**: p. 107-22.

[260] Zasowski, T., et al., *UWB for noninvasive wireless body area networks: Channel measurements and results*, in *IEEE Conference on Ultra Wideband Systems and Technologies*. 2003, Washington, DC, U.S.A.: IEEE Computer Society Press. p. 285-289.

[261] Jovanov, E., et al., *A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation.* Journal of NeuroEngineering and Rehabilitation, 2005. **2**(1): p. 1-10.

[262] Simon, J.W., et al., *Biomaterialbanken - Rechtliche Rahmenbedingungen*. Schriftenreihe zur Telematik in der medizinischen Forschung. 2006 Berlin: Medizinisch Wissenschaftlichen Verlagsgesellschaft. 240.

[263] Demchenko, Y., L. Gommans, and C.d. Laat, *Extending role based access control model for distributed multidomain applications* in *Proceedings of the IFIP TC-11 22nd International Information Security Conference (SEC 2007)*. 2007, Boston: Springer. p. 301-312.

[264] Pereira, A., V. Muppavarapu, and S. Chung, *Managing role-based access control policies for grid databases in OGSA-DAI using CAS.* Journal of Grid Computing, 2007. **5**(1): p. 65-81.

[265] Kudo, M. and S. Hada, *XML document security based on provisional authorization*, in *Proceedings of the 7th ACM Conference on Computer and Communications Security*. 2000, New York: ACM Press. p. 87-96.

[266] Gorodetski, V.I., L.J. Popyack, and V.A. Skormin, *Computer network security: Second International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security, MMM-ACNS 2003*. Lecture Notes in Computer Science Vol. 2776. 2003, Berlin; New York: Springer. xiv, 470 p.

[267] Gabbay, D.M., I. Hodkinson, and M. Reynolds, *Temporal logic: mathematical foundations and computational aspects*. Oxford logic guides; 28, 40. 1994, Oxford, U.K.: Clarendon Press.

[268] Rabinovich, A., *On translations of temporal logic of actions into monadic second-order logic.* Theoretical Computer Science, 1998. **193**(1-2): p. 197-214.

[269] Zhang, X., S. Oh, and R. Sandhu, *PBDM: a flexible delegation model in RBAC*, in *Proceedings of the eighth ACM symposium on Access control models and technologies*. 2003, New York, NY, U.S.A.: ACM Press. p. 149-157.

[270] Zhang, L., G.-J. Ahn, and B.-T. Chu, *A rule-based framework for role based delegation*, in *Proceedings of the sixth ACM symposium on Access control models and technologies*. 2001, New York, NY, U.S.A.: ACM Press. p. 153-162.

[271] Barka, E. and R. Sandhu, *Framework for role-based delegation models*, in *Proceedings of the 16th Annual Computer Security Applications Conference*. 2000, Washington, DC, U.S.A.: IEEE Computer Society Press. p. 169-176.

[272] Mokhtari, Y. and S. Merz, *Animating TLA specifications*, in *Proceedings of the 6th International Conference on Logic Programming and Automated Reasoning*. 1999, Berlin; New York: Springer. p. 92-110.

[273] Hull, R., et al., *E-services: a look behind the curtain*, in *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems*. 2003, New York, NY, U.S.A.: ACM. p. 1-14.

[274] Tefanescu, A., *Automatic synthesis of distributed systems*, in *Proceedings of the 17th IEEE international conference on Automated software engineering*. 2002, Washington, DC, U.S.A.: IEEE Computer Society Press. p. 315-320.

[275] Martinelli, F., *Keynote Talk: A Model for Usage Control in GRID Systems*, in *First International Workshop on Security, Trust and Privacy in Grid Systems*. 2007: Nice, France.

[276] Mei, Y., et al., *UCGS: A usage control approach for grid services*, in *Proceedings of the Computational Intelligence and Security Workshops*

(CISW 2007). 2007, Washington, DC, U.S.A.: IEEE Computer Society Press. p. 486-489.

[277] Pu, F., et al., *Pervasive Computing Context Access Control Based on UCON_ABC Model*, in *Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia*. 2006, Washington, DC, U.S.A.: IEEE Computer Society Press. p. 689-692.

[278] Zhang, X., et al., *A usage-based authorization framework for collaborative computing systems*, in *Proceedings of the eleventh ACM symposium on Access control models and technologies*. 2006, New York, U.S.A.: ACM Press. p. 180-189.

[279] Zhang, X., et al., *Toward a Usage-Based Security Framework for Collaborative Computing Systems.* ACM Transactions on Information and System Security, 2008. **11**(1): p. 1-36.

[280] Jensen, J., D. Spence, and M. Viljoen, *Grid single sign-on in CCLRC (Council of the Central Laboratory of the Research Council)*, in *Proceedings of the UK e-Science All Hands Meeting 2006 (AHM2006)*. 2006, Edinburgh, U.K.: National e-Science Center. p. 273-280.

[281] Drinkwater, G. *MyProxy upload tool*. [Accessed 2008 February 08]; Available from: http://tiber.dl.ac.uk:8080/myproxy/.

[282] Drinkwater, G. and S. Sufi, *Data mining using the data portal for the NESSI project*, in *Proceedings of the 2005 International Conference on Grid Computing and Applications (GCA'05)*. 2005, CSREA Press. p. 161-167.

[283] Natis, Y.V., *Service-oriented architecture scenario*. 2003, Gartner, Inc., Stanford, U.S.A. p. 1-6.

[284] Foster, I., *Service-Oriented Science.* Science, 2005. **308**(5723): p. 814-817.

[285] Thompson, J., et al., *Predicts 2006: the strategic impact of SOA broadens*. 2005, Gartner, Inc. p. 6.

[286] Windley, P.J. and G. Greif, *SOA governance: rules of the game.* InfoWorld.com, January 23 2006: p. 29-35.

[287] Weill, P. and J.W. Ross, *IT governance: how top performers manage IT decision rights for superior results*. 2004, Boston, U.S.A.: Harvard Business School Press. xiv, 269 p.

[288] IT Governance Institute, *IT Governance Frameworks*, in *IT Governance Roundtable*, P. Williams, Editor. 2008, Rolling Meadows, Ill., U.S.A.:IT Governance Institute. p. 1-12.

[289] Bioinformatics grid application for life science, *BioinfoGRID White Paper*, L. Milanesi, Editor. 2008, Institute for Biomedical Technologies (ITB - CNR): Rome, Italy. p. 1-45.

[290] Holzmann, G.J., *The spin model checker: primer and reference manual*. 2004, Boston, MA: Addison-Wesley Professional. XII, 596 p.

[291] Alur, R., et al., *MOCHA: modularity in model checking*, in *Proceedings of the 10th International Conference on Computer Aided Verification*. 1998, Berlin, New York: Springer. p. 521-525.

[292] Alur, R. and T.A. Henzinger, *Reactive Modules.* Formal Methods in System Design, 1999. **15**(1): p. 7-48.

[293] Mohammed, Y., *"Creation of players"*. June 7, 2008, Email communication with Rajeev Alur: Göttingen, Germany.

[294] Mohammed, Y., et al., *MediGRID resource usage policy - Phase 1 - development*. 2007, Department of Medical Informatics, University of Göttingen. p. 1-14.

[295] Sweeney, L., *k-anonymity: a model for protecting privacy.* International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 2002. **10**(5): p. 557-570.

[296] Yu, T. and S. Jajodia, *Secure data management in decentralized systems*. Advances in information security ; 33. 2007, New York: Springer. VIII, 462 S.

[297] Malin, B. and L. Sweeney, *How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems.* Journal of Biomedical Informatics, 2004. **37**(3): p. 179-192.

[298] Malin, B.A., *Protecting genomic sequence anonymity with generalization lattices.* Methods of Information in Medicine, 2005. **44**(5): p. 687-692.

[299] Leibniz-Rechenzentrum München. *Request for use of the high-performance computers at LRZ*. 2008 [Accessed 2008 August 26]; Available from: www.lrz-muenchen.de/services/compute/antrag/.

[300] Hoheisel, A., *Grid workflow execution service - dynamic and interactive execution and visualization of distributed workflows* in *Proceedings of Cracow Grid Workshop 2006. Vol.II*. 2007, Cracow, Poland: Academic Computer Centre CYFRONET. p. 13-24.

[301] Yavatkar, R., D. Pendarakis, and R. Guerin, *A framework for policy-based admission control (RFC 2753)*. 2000, The Internet Engineering Task Force (ITEF). p. 1-20.

[302] Lorch, M., et al., *First experiences using XACML for access control in distributed systems*, in *Proceedings of the 2003 ACM workshop on XML security*. 2003, New York, U.S.A.: ACM Press. p. 25-37.

[303] Pernul, G. and T. Priebe, *Authorization and access control*, in *European Intensive Programme on Information and Communication Systems Security 2005 (IPICS '05)*. 2005, Chios, Greece: University of the Aegean.

[304] He, H. *What Is Service-Oriented Architecture*. O'REILLY xml from the inside out. September 30, 2003 [Accessed 2008 February, 20]; Available from: www.xml.com/pub/a/ws/2003/09/30/soa.html.

List of Figures

List of Tables

Curriculum Vitae

Yassene Mohammed

Personal Information

Date of Birth	03.08.1979
Place of Birth	Kuwait city, State of Kuwait
Nationality	Syrian

Education

Since May 2005	Georg-August-University of Göttingen, Faculty of Mathematics, Göttingen, Germany PhD student in Medical Informatics
2002 - 2004	University of Applied Sciences and Arts, Department of Sciences and Technology, Göttingen, Germany Five Semesters (part time) International Master in Optical Engineering / Photonics Graduation project: “A Finite Element Method Model to Simulate Laser Interstitial ThermoTherapy in Anatomical Inhomogeneous Regions”
1997 - 2002	University of Damascus, Faculty of Mechanical and Electrical Engineering, Damascus, Syria Ten Semesters (full time) License (Licencié) degree course in Biomedical Engineering Graduation project: “Studying of Human Brain Dynamics Using Independent Component Analysis (ICA)”
1994 -1997	Secondary school
1991 -1994	Preparatory school
1986 –1991	Elementary school

Work experience

- Since October 2008, Scientific Assistant, Leibniz University of Hannover, Regional Computer Center of Lower Saxony (RRZN)

- May 2005 – September 2008, Scientific Assistant, Georg-August-University of Göttingen, Department of Medical Informatics
- January 2003 - April 2005, Graduate Assistant, Georg-August-University of Göttingen, Department of Medical Informatics