Specification and Verification of an Attribute-based Usage Control Approach for Open and Dynamic Computing Environments

Christos Grompanopoulos
Department of Applied Informatics
University of Macedonia

This dissertation is submitted for the degree of
Doctor of Philosophy

July 2014

This Dissertation Advisory Committee (DAC) consists of the following members (listed alphabetically):

- Professor Athanasios Manitsaris

  (Dept. of Applied Informatics, University of Macedonia, Greece.)

- Assistant Professor Christos Georgiadis

  (Dept. of Applied Informatics, University of Macedonia, Greece.)

- Associate Professor Ioannis Mavridis (PhD Supervisor)

  (Dept. of Applied Informatics, University of Macedonia, Greece.)

This Dissertation Defence Committee (DDC) consists of the following members (listed alphabetically):

- Associate Professor Alexandros Chatzigeorgiou

  (Dept. of Applied Informatics, University of Macedonia, Greece.)

- Assistant Professor Christos Georgiadis

  (Dept. of Applied Informatics, University of Macedonia, Greece.)

- Professor Dimitris Gritzalis

  (Dept. of Informatics, Athens University of Economics & Business, Greece.)

- Associate Professor Ioannis Mavridis

  (Dept. of Applied Informatics, University of Macedonia, Greece.)

- Assistant Professor Panagiotis Katsaros

  (Dept. of Informatics, Aristotle University of Thessaloniki, Greece.)

- Assistant Professor Panayotis Fouliras

  (Dept. of Applied Informatics, University of Macedonia, Greece.)

- Associate Professor Theodoros Kaskalis

  (Dept. of Applied Informatics, University of Macedonia, Greece.)

I would like to dedicate this dissertation to my family,

for their love, endless support and encouragement.

Acknowlegements

First and foremost, I would like to express my sincere appreciation and gratitude to my supervisor, Associate Professor Ioannis Mavridis. His guidance and support all these years and during the completion of this dissertation have been invaluable. Even in difficult times, a meeting with him always reinvigorated my enthusiasm and raised my spirits.

I am also extremely grateful to the members of my dissertation committee, Professor Athanasios Manitsaris and Assistant Professor Christos Georgiadis, for their time and dedication in reviewing my work and for their advice throughout my dissertation.

I would also like to thank my colleague and friend Antonios Gouglidis at the Department of Applied Informatics of the University of Macedonia, for sharing his thoughts and feelings during the completion of this laborious but very rewarding journey.

Finally, I would also like to take this opportunity and express my profound gratitude to my parents and all of my family for their continuous support and encouragement.

Abstract

Computing systems are always evolving in order to keep pace with the requirements posed by their users. For example, widespread of networking technologies has resulted in the transition from the standalone computer model to the client-server computer model. Moreover, social trends as frequent travelling and social networking, led to the creation of small-sized computing devices that are equipped with sensors to recognize their surrounding environment and adapt accordingly to it. Therefore, novel computing environments were emerged, as ubiquitous computing, grid - cloud computing systems and the Internet of Things. The aforementioned computing environments set new requirements in several areas, including the very important area of security. Specifically, from a computer security, and particularly from an access control point of view, the evolution of computing systems induces the creation of Open and Dynamic Computing Environments (ODCE). These environments are characterized as open since there are no boundaries between the legitimate and illegal users of a system. And they are characterized as dynamic since their configuration is constantly changing. Nevertheless, existing access control approaches are not designed for application in ODCE, therefore they cannot fully support the unique requirements posed by ODCE. Consequently, an exhaustive requirements analysis regarding access control models that are targeted to ODCE is required, which will subsequently help in the definition of proper access control approaches for application in ODCE.

In this dissertation, we analyze existing access and usage control approaches to identify a number of unique requirements posed by ODCE. Secondly, we formally define an attribute

based usage control model for ODCE that is designed based on the identified requirements. Last but not least, we check the proposed model for its correctness, i.e., the adherence of the model to its initially defined specifications.

Specifically, we provide information on the RBAC, ABAC and UCON access/usage control models, which are mostly applicable in the examined case of ODCE. Moreover, we present information regarding the modeling of concurrent systems, as the access control systems. The aforementioned information helps in the identification of the demanded specifications in the context of access/usage control models in ODCE. Moreover, state-of-the-art verification techniques are described since it is required to check the correctness of the newly defined models in respect to their initial specifications. In turn, we highlight through representative usage scenarios the challenging issues and limitations that are introduced when attempting to utilize the UCON family of models in modern computing environments.

Based on the requirement analysis results of access control in ODCE, we propose a formal specification of UseCON model using the Temporal Logic of Actions (TLA+) formal language. Consequently, we evaluate UseCON, which incorporates a number of significant features compared with existing access/usage control models. The proposed model, firstly, results in support of extended expressiveness over the existing usage control models. This extended expressiveness of UseCON stems from the utilization, during the creation of usage allowance decision, of information originating from either a single or a set of both direct and indirect entities. Secondly, UseCON inherently supports the utilization of historical information of usages through the automatic management of use entities. Additional advantages of the proposed model are considered to be its support for simplified policy administration processes, and additional unique features like the negotiation of actions etc.

In order to prove the correctness of the defined UseCON model we contacted research on formal verification techniques, thus, resulting in the definition of a set of safety and liveness properties. The verification of the defined set of properties was performed by applying

a model checking technique with the TLC model checker. We conclude this dissertation by providing additional information regarding the TLC verification process, and further discuss our findings, pointing to the contributions of this research work and to possible future research directions, as well.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Technological innovations are constantly transforming the architecture of computer systems. For example, the widespread of networking technologies resulted in the transition from the standalone computer model to the client-server computer model. Moreover, social trends like frequent travelling and social networking, led to the creation of small-sized computing devices (e.g., PDA's) that are equipped with sensors to recognize their surrounding environment and adapt accordingly to it. Consequently, novel computing environments are emerged, as ubiquitous computing (Weiser, 1994, 1999), Internet of Things (Atzori et al., 2010), Cloud (Foster et al., 2008) and Grid computing environments (Foster et al., 2001). From a computer security, and particularly from an access control point of view, the evolution of computing systems induces the creation of Open and Dynamic Computing Environments (ODCE) (Kagal et al., 2006). ODCE are characterized as open since there are no boundaries between the legitimate and illegal users of a system. This occurs since any user without clear knowledge of her identity may request access to a system's resource. ODCE are also characterized as dynamic since the configuration of these systems is constantly changing (e.g., users join in or leave, request or terminate access to resources). Moreover, the utilization of contextual information for the creation of an access decision enhances the

dynamic nature of ODCE. In the rest of this chapter, we elaborate on the clarification of the motivation of this research work, along with the objectives and the areas that we conducted research upon.

1.1 Motivation

The evolution of computing systems introduces new requirements for the access control systems that are called to support them. In traditional computing systems operated by a small number of users, access control models utilize only a single criterion for allowing an access request. This criterion is related to the security characteristics of the subject and object involved in the requested access (Samarati and Vimercati, 2001a). Specifically, whenever a subject requests to access an object, the subject's clearance and object's classification when it comes in Mandatory Access Control (MAC) models (Sandhu, 1993), the subject's identity when it comes in Discretionary Access Control (DAC) models (Qiu et al., 1985) and an activated role from a set of authorized roles of a subject when it comes in Role Based Access Control (RBAC) models (Sandhu et al., 1996), are utilized accordingly. In contrast with MAC, DAC and RBAC, attribute based access control approaches (Jin et al., 2012) are capable of supporting partially-unknown users, by utilizing not the identity but instead a number of subject and object security related characteristics, which are expressed in the form of attributes. Moreover, a number of extensions to the aforementioned access control models have been proposed that are able to utilize contextual information for the creation of an access decision (Kulkarni and Tripathi, 2008; Kumar et al., 2002).

The Usage CONtrol (UCON) family of models (Park and Sandhu, 2004) provides an integration of traditional access control, digital rights and trust management. Additionally, UCON introduces the concepts of continuity of decision and attribute mutability. Continuity of decision considers that access to a resource is no longer a spontaneous action, but it may

last for some time. Consequently, decision factors are evaluated not only before, but also during the exercise of an access, and thus, resulting into evolving the concept of access control to that of usage control. Moreover, attribute mutability is responsible for updating attribute values of the participating entities (i.e., subject or object) as a side effect of an allowed usage. Consequently, new permissions are created due to these updated attribute values, and thus, results in a dynamic usage control model. Moreover, UCON utilizes three criteria for the creation of a usage decision. These are the security related characteristics, (i.e., properties) of an entity, contextual and historical information about usages. Entity's properties, which are expressed into its attribute values, are utilized in authorizations, while contextual information represented by special system variables is evaluated in conditions. Historical information of usages is utilized either by authorizations with attribute mutability or by obligations. The aforementioned obligations are actions performed by subjects in the system in order to permit the allowance of a usage request.

However, despite of its aforementioned virtues, UCON raises a number of challenging issues in modern computing environments (Grompanopoulos and Mavridis, 2012b). Specifically, whenever a subject s requests the execution of a right on an object o then only the decision criteria related directly with s and o are utilized in the decision creation process (Park and Sandhu, 2004; Zhang et al., 2005). These entities are called *direct* entities of the usage. However, it's possible that decision criteria related with other entities than s and o to be also evaluated for the creation of the usage decision. These entities are called *indirect* entities. For example, let's assume a scenario where Alice is a child. If Alice requests to visit a museum then security criteria related with Alice's father (i.e., Bob) might be utilized for the creation of the access decision. A criterion of this category might be the fact that Bob is the owner of a family free entrance card. Another limitation imposed by UCON is its inability to support complicated operations of subjects on objects required by modern computing environments. For example, every banking transaction incorporates

detailed information including a timestamp, the currency, the transaction's purpose and so on. In UCON, it is not feasible to express the previous transaction using only a simple right (Grompanopoulos and Mavridis, 2012b). This results in the creation of an enormous amount of UCON rights by the usage control system. Moreover, the attribute mutability mechanism introduced in UCON presents a number of limitations in utilizing historical information of usages. Specifically, attribute mutability in UCON records only the permitted usages and not the denied ones. Additionally, UCON attribute mutability does not distinguish between usages terminated by an ongoing rule violation and those terminated by a subject request.

Therefore, the motivation of this dissertation is three-folded. Firstly, we concentrate on the on the analysis of the access control requirements that are posed by ODCE, and on the evaluation of existing approaches in access control. Secondly, in order to support the new requirements imposed by ODCE, a Use-based usage CONtrol model (UseCON) was proposed in (Grompanopoulos and Mavridis, 2012a; Grompanopoulos et al., 2013). UseCON, similarly to UCON, is able to utilize during the decision making process the three aforementioned criteria (i.e., properties, context and historical information). Additionally, UseCON is able to utilize criteria originating from either *direct* or *indirect* entities. Moreover, it supports complicated operation modes of subjects on objects (e.g., banking transactions) and provides a detailed utilization of historical information of usages compared with the one provided by the attribute mutability mechanism in UCON. Last but not least, a formal specification of the UseCON model is provided and a set of properties are defined so that to help in the formal evaluation of the proposed attribute based usage control model using model checking techniques.

1.2 Objectives

Attribute Based Access Control (ABAC) and usage control, with UCON being a dominant approach in usage control models, have useful characteristics for the application of the access control process in ODCE. Specifically, ABAC represents a point on the spectrum of logical access control from simple access control lists to more capable role-based access, and finally to a highly flexible method for providing access based on the evaluation of attribute (Hu et al., 2014). UCON mediates access to resources not only before, but also during the access, and thus, leverages the concept of access control to that of usage control (Park and Sandhu, 2004). However, there are cases were more expressiveness and functionality is required, as in the case of the examined ODCE. Although a considerable amount of work has been done in the field of access control for ODCE (see Chapter 2), access control models that are available to date are not able to fully support the unique characteristics of ODCE (see Chapter 3). In this dissertation, we focus on the definition of an attribute based usage control approach that is able to cope with the requirements posed by ODCE. The main objective of our work can be summarized as follows.

In the context of requirements analysis we identify a set of requirements that must be fulfilled by ODCE through a series of use cases and related literature. Secondly, we define an attribute based usage control model that will integrate the required functionality in respect to the identified requirements. Finally, the last objective is to provide a formal method to verify the correctness of the defined access control model.

1.3 Research areas

The contributions of this dissertation are three-fold; first, we analyze the access control requirements of ODCE and evaluate the limitations and shortcomings of existing access

and usage control approaches in order to identify a number of unique characteristics posed by ODCE. Secondly, we formally define an attribute based usage control model for ODCE that is designed based on the identified requirements. Last but not least, we check the proposed model for its correctness, i.e., the adherence of the model to its initially defined specifications. In the remainder of this section, we present the contributions in more details.

1.3.1 Requirement analysis

Through representative example scenarios and related literature, we highlight the additional requirements that are posed when attempting to implement access control process in ODCE. Moreover, we investigate the limitations of existing access control solutions (UCON) when they are applied to ODCE. The access control requirements that are gathered, together with the limitations of the UCON model, are utilized as the design characteristic for the proposed attribute based usage control model.

1.3.2 Specifications modeling

An important contribution of this dissertation is the definition of an attribute based usage control model for ODCE. Specifically, the original contributions of our work can be summarized as follows.

The UseCON model incorporates a number of significant features compared with existing access/usage control models. Firstly, UseCON's extended expressiveness over the other models is the result of utilizing information originating from either a single or a set of both *direct* and *indirect* entities in the creation of the usage allowance decision. Secondly, UseCON inherently supports the utilization of historical information of usages through the automatic management of use entities; therefore, a policy administrator via the definition of policy update procedures, does no longer require further to record usage information.

Consequently, there is a clear distinction between the model's operations, (i.e., utilization of historical usages) and the operations performed by an administrator (i.e., definition of policy rules).

1.3.3 Formal verification

The third contribution of this dissertation refers to the formal verification of the UseCON model. Therefore, we provided a sound and solid formal definition of the internal actions performed by the model through the development of a formal specification for the model. This specification provides an unambiguous and sound understanding of the novel concepts introduced by UseCON (i.e., the utilization of the *indirect* entities in the usage control decision and the extended utilization of historical information of usages, through use entities). Moreover, the support of concurrent operations by multiple usages differentiates the specification from existing usage control specifications. In addition, a number of properties (i.e, type consistency, *safety* and *liveness*) for both the core model and policy implementations were verified for their correctness (i.e., adherence of the defined model and supported policies to the initially defined requirements) through the application of an automated and error-free model checking technique.

1.4 Structure of the dissertation

This chapter has discussed the motivation and the main objectives of our work and outlined the major contributions of this dissertation. The remaining chapters are structured as follow.

Chapter 2 discusses background information regarding the operational characteristics of ODCE focusing in the impact on system's security. Moreover, existent solutions in the area of access control models are presented. Furthermore, the specification and verification fundamentals, especially in concurrent systems, are presented with a focus in the utiliza-

tion of model checking techniques for the verification of system properties. In addition, background information regarding the Temporal Logic of Actions (TLA+), a system specification and verification language, are also presented. Finally, the chapter concludes with a presentation of related works on the specification and verification of existent access control models.

Chapter 3 presents a requirement analysis regarding access control in ODCE through several examined scenarios and related literature. Moreover, a detailed critique on the utilization of the usage decision criteria by existing usage control solutions, specifically UCON, is also performed. The results of the previous procedure are used for the determination of the design characteristics for the usage control model proposed in Chapter 4.

Chapter 4 describes the proposed attribute based usage control model for ODCE. The chapter begins with a brief description of the proposed Use-based usage control model (UseCON) and of it's core features. In turn, the formal specification in TLA+ of UseCON is presented next. Furthermore, the chapter concludes with a formal specification in TLA+ of a policy example. Specifically, two different specification approaches are presented that implements the same "high-level" policy.

Chapter 5 evaluates the proposed model in terms of the presented characteristics and properties. Specifically, Section 5.2 highlights the novel characteristics presented by UseCON as the abstraction of actions and the utilization of historical information. Moreover, Section 5.3 defines the type consistency, safety and liveness properties for the use management process of the UseCON specification, as this was presented in Chapter 4, in order to be checked for correctness. In addition, analogous properties are defined for two specification approaches of a policy example that was presented in Chapter 4.

Chapter 6 presents information on the verification process of the UseCON properties already defined in Chapter 5 in a TLA+ enabled model checker software. Furthermore, a discussion is presented regarding the implementation metrics and performance results of the

applied model checking technique.

Chapter 7 summarizes the contributions of this dissertation and outlines future work.

Chapter 2

Background

2.1 Introduction

This chapter provides information on access control in ODCE. Specifically, the dominant access control approaches are presented along with their specification and verification. The process of access control in a computing system ensures that only authorized users have access to the resources of the computing system (Capitani di Vimercati et al., 2007; Gollmann, 1999). Access control is an essential requirement for a system in order to provide the properties of confidentiality, integrity and availability. The permissions or authorizations of users are calculated with policies or rules (Samarati and Vimercati, 2001b). The remainder of the chapter is organized as follows. Section 2.2 presents security threats in modern computing environments. Section 2.3 provides information on access control, and elaborates on three access control approaches i.e., RBAC, ABAC and UCON. RBAC gains popularity due to the large number of installed systems and the easiness of the administration that offer. On the other hand ABAC approaches provide more flexible access control due to the fact that they can base the access control decision to a number of criteria and not only the identity of the participating entities (subject, object). Moreover, another interesting access/usage con-

trol model (UCON) that is based on top of an ABAC model is also presented. In Section 2.4 we present information on the system's specification with a focus on transition systems and system's verification with focus on model checking. Consequently, in Section 2.5 we provide further information regarding the specification and verification of systems in the TLA+ language. UCON design characteristics as the utilization of contextual information and continuity of decisions renders it a perfect candidate for use in ODCE. Therefore, in Section 2.6, we elaborate on the specifications and verification solutions that have been developed for the UCON model.

2.2 Open and Dynamic Computing Environments

The proliferation of computing into the life of millions of people created the need for novel computing environments. Ubiquitous computing systems have natural interfaces that are capable of supporting common forms of human expression and leverage more of our implicit actions in the world (Abowd and Mynatt, 2000; Weiser, 1994). Moreover, contextual information is utilized by applications in order to provide relevant information or services to the user (Abowd et al., 1999; Chen et al., 2000). Pervasive computing environments (Kagal et al., 2001; Saha and Mukherjee, 2003) describe the interaction, coordination, and cooperation of numerous, casually accessible, and often invisible computing devices and services. In a try to eliminate the gap between information and natural objects, the Internet of Things is defined as a superset of all objects that are uniquely identifiable by electromagnetic means and for which it is possible to specify a semantic and/or behavior (Ashton, 2009; Atzori et al., 2010). The Grid (Berman et al., 2003) and more recently the Cloud (Foster et al., 2008; Theoharidou et al., 2013) computing environments represent large scale distributed computing environments that deliver on-demand to the users computing power, storage, platforms and services. A key feature in Cloud and Grid computing is that the

supplied computing power, storage or services are abstracted, virtualized and dynamically scaled and managed Foster and Kesselman (2003).

All the aforementioned computing environments extend the security perimeter from an "isolated" environment, as a desktop computer, to a more widely defined and dynamically modified environment, as a mobile device connected to the Internet through a wireless connection. Moreover, the operations performed by users on their mobile devices varies from simple, e.g., checking email, controlling home devices, to more complex, e.g. booking airline tickets buying or selling good, manage bank accounts. Consequently, the violation of security in modern computing environments may have an impact not only on business, but also in the personal data of the users (Gritzalis et al., 2014). Therefore, information system's security and privacy, once narrow topics primarily of interest to IS designers, have become critically important to society at large. People are the security administrators of their own data and require having control of data disclosure and of further their use. Finally, computing systems are evolving to distributed systems that are characterized as Open and Dynamic Computing Environments (ODCE). Open in that they don't pre-identify a set of known participants, and dynamic in their configuration in modified constantly because user's login/logout on a high frequent basis and also because contextual information, which is utilized for the access decision, is constantly changing.

2.3 Access control models

Access control is the security process of a computing system that mediates every request to resources and data and determines whether the request should be granted or denied (Benantar, 2005; Gollmann, 1999). Access control is not an isolated process, but it relies on and coexists with other security services in a computer system like authentication and audit (Sandhu and Samarati, 1996). The development of an access control system requires the

specification of three abstractions of controls: access control policies, models, and mechanisms (Benantar, 2005; Hu et al., 2006; Samarati and Vimercati, 2001b). Access control policies are high-level requirements that specify the rules who are allowed to access which resources under certain conditions. The policies are enforced through access control mechanisms that are low level, software and hardware, functions. Rather than evaluating and analyzing policies at a mechanism level, models are defined that are formal representations of the access control policies and are useful for proving theoretical limitations of a system. Moreover, the fundamentals functions of Access Control Decision (ADF) and Enforcement (AEF) was proposed in (ITU-T, 1995). ADF is a specialized function that creates access control decisions by applying policy rules to an access request and the surrounding context. Moreover, the AEF is a specialized function that is part of the access path between the requestor and the resource that enforces the decision created by the ADF.

Therefore, various access control policies and models have been introduced Sandhu and Samarati (1994), namely the Mandatory Access Control policies (MAC), the Discretionary Access Control policies (DAC) and the Role Based Access Control policies (RBAC), and lately the Attribute Based Access Control (ABAC). Each one of them serves specific security requirements in different working environments. Furthermore, research on the MAC, DAC, RBAC and ABAC has proven that an access control model, which can express the RBAC policies is also capable of enforcing both MAC and DAC policies (Ferraiolo et al., 2003, chap. 6), and moreover that ABAC can express RBAC policies. The ABAC model was mainly introduced to overcome a number of RBAC's shortcomings (Yuan and Tong, 2005). Following in this section, we provide information on the standard for the RBAC (ANSI, 2004), ABAC and Usage Control (Park and Sandhu, 2004; Sandhu and Park, 2003; Zhang et al., 2008). UCON is stated in this section since it is an ABAC approach able to cope with usage control. In turn, usage control seems to fulfill several requirements posed by ODCE.

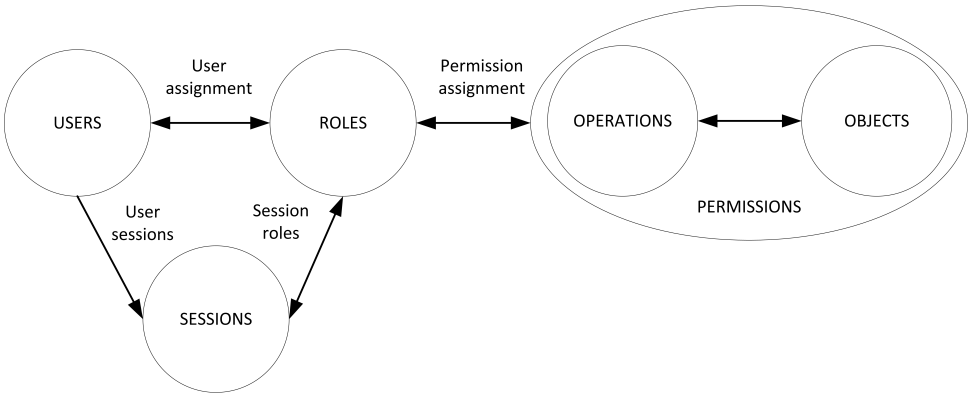

Fig. 2.1 The core RBAC model (Ferraiolo et al., 2003).

2.3.1 Role based access control (RBAC)

The RBAC model has received considerable attentions from researchers for its capabilities of abstraction and generalization. RBAC is considered to be abstract since it includes only properties that are relevant to security. Furthermore, it is considered to be a generic model, because many designs could be considered valid interpretations of the model (Ferraiolo et al., 2003). In addition, RBAC supports various access control principles as Least Privilege, and Separation of Duties (SoD)/Administrations (Sandhu et al., 1996). The RBAC model consists of four components having each a different functionality. The components are Core RBAC, Hierarchical RBAC, Static Separation of Duty (SSD), and Dynamic Separation of Duty (DSD).

The core RBAC model consists of five static elements, as depicted in Figure 2.1: users, roles, and permissions, which contain operations and objects. Following Figure 2.1 it applies that roles are assigned to users and permissions are assigned to roles. The mapping between users and roles is considered to be of type many-to-many (i.e., one user can be assigned to many roles and many users can be assigned to one role). The same applies to the role to permission assignment. It is noteworthy to state that negative permissions are not supported in RBAC. Furthermore, the indirect assignment of users to permissions greatly enhances the administration in RBAC, and revocation of assignments can be done easily.

Moreover, in RBAC implementations we can distinguish the design and run-time phases. During the design time phase, system administrators can define assignments between the elements, and that model enforces the aforementioned assignments during the run-time phase.

The run-time phase is achieved by the concept of the session. This unique, among other group-based access control mechanisms, feature allows a set of users' roles to be activated. This means a user could be assigned to various roles during the design phase, but do not need to be always or simultaneously activated (by the principle of least privilege). However, the capability of sessions has been questioned with a suggestion of replacement for them as stated in (Li et al., 2007).

The Hierarchical RBAC enhances administration flexibility through the capability of permission (operations to objects) inheritance; permissions (assigned to a role) can be inherited to another role through hierarchical relation assignments without reassigning the same permissions to the inherited role. For instance, assuming two roles r_1 and r_2 and two permission sets $PRMS_1 = (p_1, p_2)$ and $PRMS_2 = (p_3, p_4)$, which are initially assigned to roles r_1 and r_2, respectively. Role r_1 inherits role r_2 means all permissions of r_2 are also available to r_1. The inherited permission can be expressed by the union of $PRMS_1$ and $PRMS_2$. The immediate inheritance relation is denoted by the $\rightarrow$, for example, $r_1 \rightarrow r_2$. User membership refers to the assignment of users to roles in a hierarchy, and thus, users are authorized to have all the permission assigned to roles either directly or via inheritance. The Hierarchical RBAC supports general and limited role hierarchies. General hierarchies are composed of partial order sets of common inheritance relations. And, in more restrictive environments, limited hierarchies require the existence of either a single immediate ascendant or descendant role in the hierarchy. Mathematically, hierarchy is a partial order defining seniority relations between roles, whereby senior roles acquire the permissions from their juniors and junior roles acquire users from their seniors (ANSI, 2004).

Another advantageous characteristic of RBAC is that it can constrain authorization with

SSD and DSD relationships to prevent the Conflict Of Interest (COI), which is common from business requirements. SSD handles the enforcement of static COI policies. For example, let r_1 and r_2 be two conflicting roles, and user u_1 assigned to role r_1. RBAC prohibits the assignment of user u_1 to role r_2 by enforcing an SSD constraint between roles r_1 and r_2 since the two roles are COI. The constraints are defined and restricted in the design phase. In the presence of role a hierarchy, the SSD constrains are enforced in the same way for all the directly assigned and inherited roles. DSD relationships can handle COI policies in the context of a session, where a user is activated with a set of assigned roles when logged into the system. As in SSD, in DSD also applies that constraints are specified in the design phase. However, they are enforced during the run-time of the authoring process through activated sessions, and thus, it prevents the simultaneous activation of two or more conflicting roles.

Moreover, one of RBAC's greatest virtues is that of role based administration. RBAC administration is divided into two spaces that of user and administrator. The former includes user roles and the latter administrative roles with permissions and operations, respectively. Once again, the principle of least privileged is maintained. In the literature we have identified several RBAC administration models that are proposed (Crampton and Loizou, 2002), (Ferraiolo et al., 2003), (Oh and Sandhu, 2002), (Sandhu et al., 1999). However, each of them is making use of a different approach in role based administration.

An area of great scientific interest, where a lot of effort has been presented, is the capability to support contextual information in RBAC. A first attempt was the introduction of environment roles in an attempt to uniformly treat contextual information with subject and object characteristics (Covington et al., 2001). A different approach was proposed in (Kulkarni and Tripathi, 2008) where the fulfillment of contextual condition are required for role admission or dynamic object binding. A similar approach that dynamically adjusts role assignments and permission assignments based on contextual information was also pro-

posed in (Zhang and Parashar, 2004). Moreover, the integration of contextual information with team based access controlwas proposed in (Georgiadis et al., 2001) in orfer to provide access control for collaborative activity best accomplished by teams of users.

2.3.2 Attribute based access control (ABAC)

The distinguishing design characteristic in ABAC is that permissions are not assigned directly to the identities of an entity (subjects-objects). Instead, the attributes of entities are used as a basis for forming authorizations. Attributes are considered to be the security-related characteristics of the participating entities (e.g., a subject's attribute could be: his/hers name, date of birth, home address, training record etc). Consequently, the process of an access control decision in an ABAC system is performed as follows: "subjects that request to perform operations on objects are granted or denied based on the subject's, object's assigned attributes, the environment conditions, and a set of policies that are specified in terms of those attributes and conditions." (Hu et al., 2014). As stated in the previous definition, a policy plays a crucial role in the operation of ABAC. A policy is created by the resource administrator or owner and it is an access control rule using attributes of subjects and objects to govern the set of allowable capabilities (e.g., all doctors with 10 years of experience and more can view the patient's medical records). Moreover, recent definitions of ABAC include environmental conditions into the policy rules, and thus, adding flexibility to the possible declared policies (Hu et al., 2014). As a consequence to the design characteristics of ABAC it provides significant improvements to the access control process. Firstly, access decisions can change between requests by simply changing attribute values, and without the need for changing the subject/object relationships that define the underlying rule sets. This provides a more dynamic access control management capability and limits long-term maintenance requirements of object protections. Secondly, ABAC enables object owners or

administrators to apply access control policy without prior knowledge of the specific subject and for an unlimited number of subjects that might require access. As new subjects join the organization, rules and objects do not need to be modified. Thirdly, ABAC solutions can easily support multi-factor decisions (e.g., decision dependent on a number of users attributes possibly combined with environmental conditions) unlike other access control model as in RBAC where the handling of analogous access decisions would require the creation of numerous roles that are ad-hoc and limited in membership, leading to what is often termed "role explosion". A high-level definition of ABAC is depicted in Figure 2.2 where the ABAC access control mechanism receives the subject's access request and then it examines the subject's and object's attributes against a specific policy. The access control mechanism then determines what operations the subject may perform upon the object.

Recent studies have proved that existing access control models like DAC, MAC and RBAC, can be "easily and naturally" configured by ABAC Jin et al. (2012). Moreover, the flexibility and scalability of an ABAC model, compared with RBAC, contributes to the creation of access control models and mechanisms for ODCE that are based on ABAC Lang et al. (2009); Wang et al. (2004).

2.3.3 Usage control (UCON)

The UCON model (Park and Sandhu, 2004) consists of six components, viz. subjects, objects, rights, authorizations, obligations and conditions. Subject is the entity that requests the usage of another entity (object), both introduced from the primary access control models (Lampson, 1974). However, subjects and objects in UCON are defined and represented with security relevant characteristics called attributes. The utilization of subject and object attributes provides the capability for fine-grained usage control on resources and the ability to support users without knowledge of their identity (Kagal et al., 2006). The usage modes

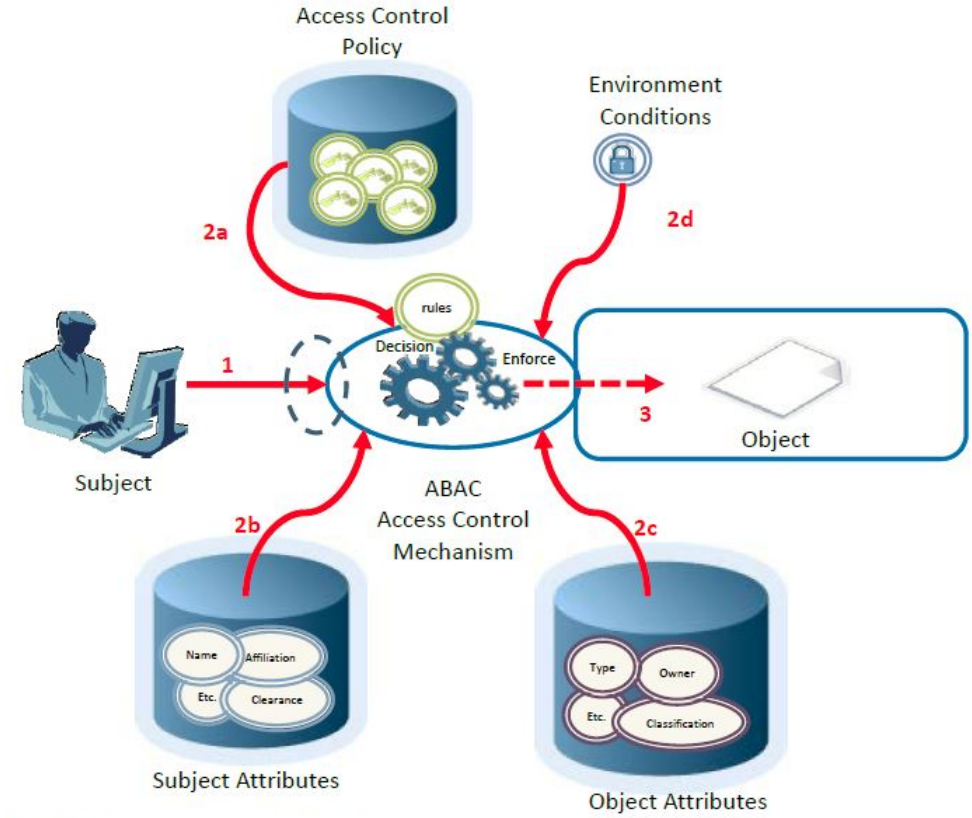

Fig. 2.2 Basic ABAC scenario (Hu et al., 2014).

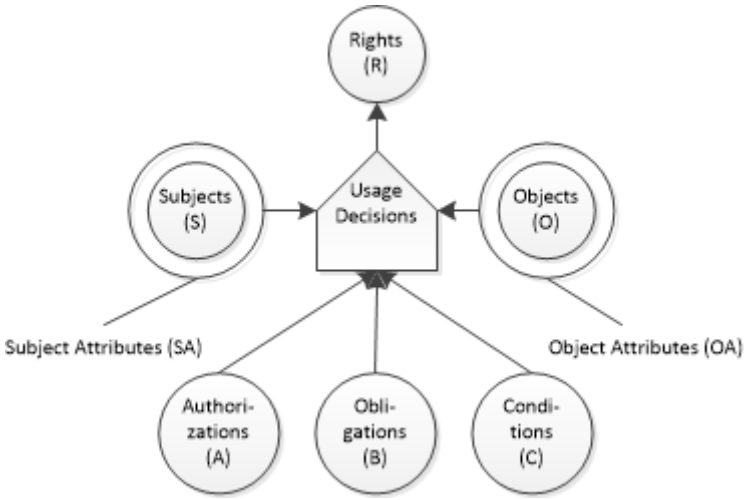

Fig. 2.3 Components of the UCON model (Park and Sandhu, 2004).

that a subject can exercise on an object are represented in UCON with rights. Rights are specific and they are not associated with attributes, in contrast to subjects and objects.

The allowance of a subject to exercise a right on an object is not predetermined, but it is decided at the time of the usage request, with the evaluation of a decision factor. UCON introduces three decision factors, named Authorizations (A), oBligations (B) and Conditions (C). Authorizations are functional predicates that utilize the values of subject and object attributes. An obligation is one or more activities that must have been performed by the subject as a requirement for the requested usage to be allowed. The selection of the required obligations is based on the attribute values of the subject and the object involved in the usage. Moreover, the subject and object pair in an operation required by an obligation, may differ from the subject and object pair involved in the usage under consideration. Conditions are also functional predicates, like authorizations, but they utilize only system related information contained in special system variables, named condition variables. The UCON components and their relationships are depicted in Figure 2.3 as originally presented in (Park and Sandhu, 2004).

Two innovative concepts introduced in UCON is continuity of decision and attribute

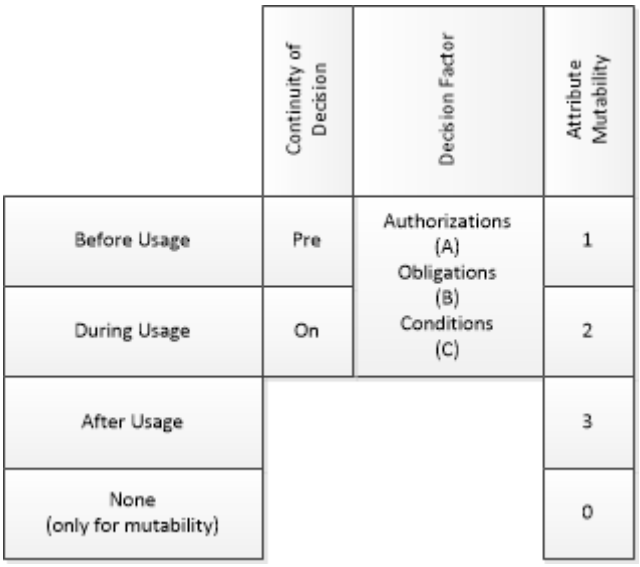

Fig. 2.4 Family of UCON sub-models.

mutability (Park and Sandhu, 2004). More specifically, the exercise of a right from a subject in an object, is not a spontaneous action, but it may last some time and include several sub-actions. With the utilization of continuity of decision, authorizations, obligation and conditions are categorized into pre and ongoing (required before and during the usage, respectively). Moreover, attribute mutability updates the attributes of the subject or the object as a consequence of an allowed usage. The attribute update procedure can be executed before (1), during (2), after (3) the exercise of an allowed usage, or not executed at all (0). Figure 2.4 is a graphical representation of the family of UCON sub-models (Park and Sandhu, 2004) that encompasses decision factors, continuity of decision and attribute mutability.

A key factor for UCON's acceptance by modern computing environments as ODCE, is the development of adequate usage control enforcement methods. Some solutions for the enforcement of usage control already exist, but to cover a wide range of application areas. These solutions need to mature and expand. This leaves many interesting areas of current and future research that are of great practical relevance (Nyre, 2011) (Pretschner et al., 2008). In addition (Almutairi and Siewe, 2011) proposes Context-Aware Usage CONtrol

(CA-UCON) model which extends the traditional UCON model to enable adaptation to environmental changes in the aim of preserving continuity of access. Moreover, additional improvements to the original UCON model have also been proposed. In (Yang et al., 2009) permissions are valid only for a set time period and only for a given number of times. Work in (Krautsevich et al., 2010) calculates the outcome of usage control decision information regarding trust and risk. A specialized solution that permits the operation of dynamic virtual organizations in GRID environment is also presented in (Gui et al., 2011)

2.4 Specification and verification of systems

In this section, we provide prerequisite information regarding the formal specification and verification of concurrent computer systems. Specifically, we provide information on a system's specification with transition systems, along with verification techniques with a focus on the specification and verification of properties using a model checking technique.

2.4.1 Specification fundamentals

Concurrent computer systems[1] are a special category of systems where many computations are executing simultaneously and possible interacting with each other (Ben-Ari, 2006). Support of concurrency is fundamental in *interactive* systems, (i.e., systems where users and computers are inter-acting), because there is no way to limit the users' behaviour or prevent them from interrupting the system's execution. The proposed attribute based usage control model is an *interactive*, and thus, a *concurrent* system since a user may arbitrarily request new usages or terminate existing ones during the execution of system's computations. In order to describe the behaviour of a system in a sound mathematical basis, formal methods are widely used in computing science (Wing, 1990). TLA+ is a formal language especially

[1] We onwards use the terms of computer system and system interchangeably.

designed to express and specify concurrent systems (Lamport et al., 2002). TLA+, as the majority of temporal logic formal approaches, uses the semantics of transitions systems, which is the dominant method for formal representation of concurrent systems.

Transition systems

Formal methods provide a framework to systematically specify, develop and verify computer systems (Wing, 1990). The first step for the creation of a system's specification is the identification of all system's *values* (i.e., every data item manipulated by the system). System's algorithms assign these *values* to a set of infinite *variables* names. The second step is the description of the system's behaviour with a *transition* system class of models which constitute the dominant method for the description of concurrent systems (Baier and Katoen, 2008). *Transition* systems are directed graphs where nodes represent system *states* and edges represent *transitions* or *actions* (i.e., state changes). A system *state* is the assignment with a *value* to every *variable var* of the system. We write $s[[x]]$ to denote the assignment of a value in *variable x* in *state s*. A *constant* is a variable that is assigned with a stable *value* to every *state* of the system. *Actions* specify how the system can evolve from one state to another. A sequence of system *states*, that is created by the application of a series of *actions* on an initial *state*, composes a *behaviour*. An example of a transition system is depicted in Figure 2.5.

2.4.2 Verification techniques

The principal methods for the verification of complex systems can be grouped under four types. These are testing, simulation, deductive verification, and model checking (Heljanko, 2006). Testing is performed on the system itself. However, testing of usage control systems for ODCE is not always a cost effective process, since it can be performed when an

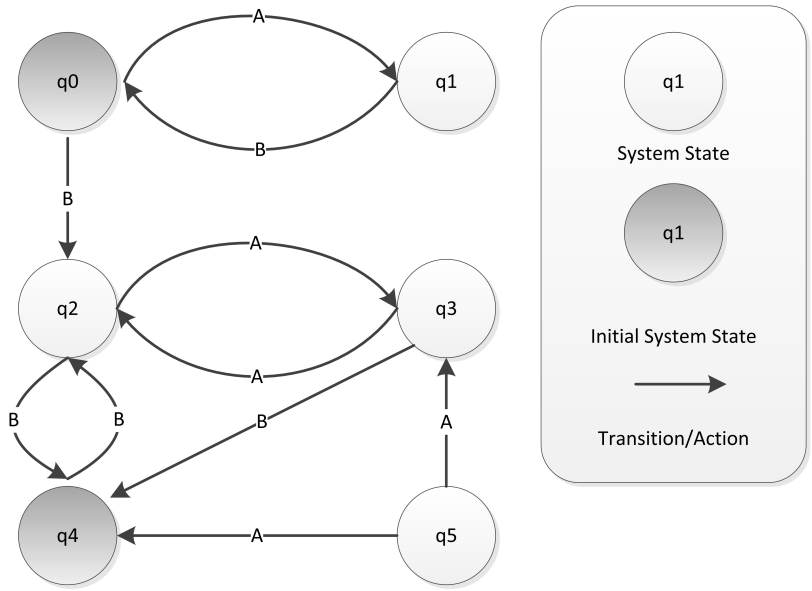

Fig. 2.5 Transition system

implementation of the system is available. Furthermore, it can only prove the existence of bugs, but not their absence. Similarly, simulation-based approaches suffer from lack of completeness as it is impossible or impractical to test all system trajectories. Furthermore, simulation-based testing is semi-automatic since the user must provide a large number of test cases. Deductive verification is based on manual mathematical proof of correctness of a model of a system. It is a very highly cost process and, furthermore, requires highly skilled personnel. Model checking verifies a system's model against defined properties; it is not vulnerable to the likelihood that an error is exposed. This contrasts with testing and simulation that are aimed at tracing the most probable defects. Additionally, it provides diagnostic information in case a property is invalidated, which is very useful for debugging purposes. In principle, model checking is an automated process and its use requires neither a high degree of user interaction nor complex test data. Furthermore, it does not require the development of custom tools for verifying a system, which can be a time-consuming and error-prone process. On the contrary, verification via model checking can be applied using existing model checkers. Therefore, model checking can serve as a technique to de-

tect non-conformance, between an access or usage control system and its initially defined specifications, as efficiently as possible.

2.4.3 Model checking

Model checking is an automated technique that, given a model of a system and a formal property, systematically checks whether this property holds for a given/any state/behavior of that model. In order to verify that a system satisfies a property the following three things are required (Huth and Ryan, 2004):

- A model of the system, which is developed in a formal specification language of the model checker. The model of a system describes all the behaviours of the system in an accurate and unambiguous way. They are mostly expressed using transition systems as described in the previous section.

- The property of the system, specified in the temporal logic language and used by the model checker. Temporal logic is basically an extension of traditional propositional logic with operators that refer to the behaviour of the systems over time. To make a rigorous verification possible, properties should be described in a precise and unambiguous manner.

- The execution of the model checker having as inputs the model of the system and the system's properties. The model checker first has to be initialized by appropriately setting the various options and directives that may be used to carry out the exhaustive verification. Subsequently, the actual model checking takes place. This is basically a solely algorithmic approach in which the validity of the property under consideration is checked in all states of the systems model.

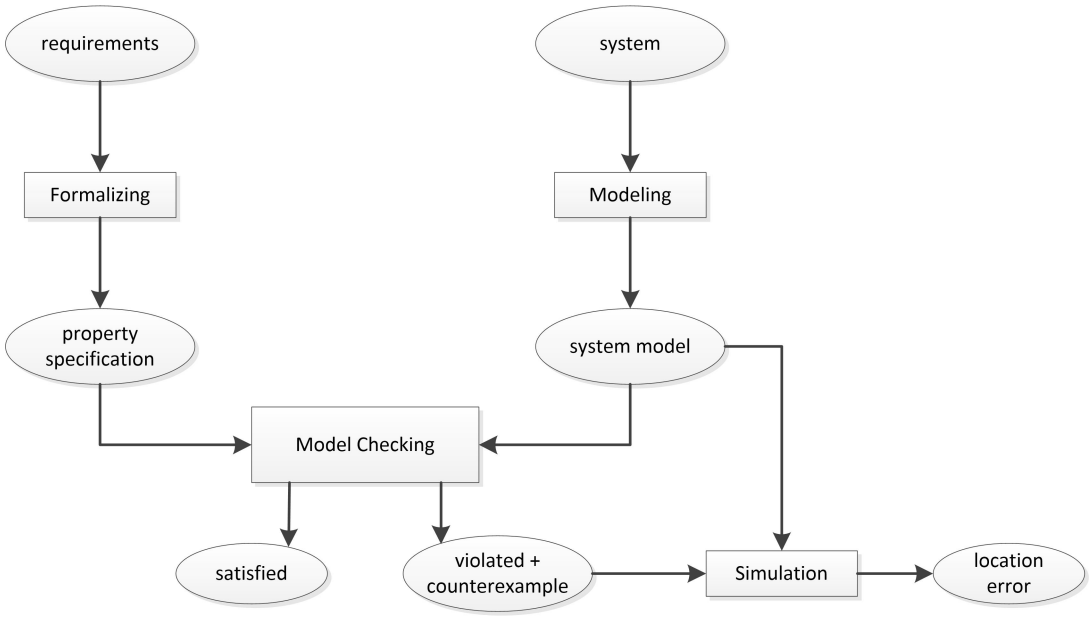

Fig. 2.6 The model checking process (Baier and Katoen, 2008)

In Figure 2.6 we depict the model checking process with all the intermediate stages. In a nutshell, both the system and requirements are formally defined in a temporal logic language. Using a model checker software, it is feasible to check the satisfiability of the defined requirements in the model. In case a property doesn't hold, then a counterexample is provided showing the full trace that led to the erroneous behaviour.

The model checking software applies algorithms for exploring the state space of a transition system to determine if it obeys a specification of its intended behavior. There may be a modeling error that implies the correction of the model and the verification has to be started with the improved model. If the error analysis shows that there is a discrepancy between the design and its model, then either a design error has been exposed or a property error has taken place. These errors when using model checking software are accompanied with counter example traces. The automatic generation of such counter example traces consists of an important tool for design and debugging purposes of the systems.

The properties that can be defined in a system are categorized, but not limited, into *safety* and *liveness*. A *safety* property determines that the system does not attempt to do something

which is not permitted, while a *liveness* property ensures that the system will eventually do something that is required (Kindler, 1994).

Some of the problems of model checking are (Baier and Katoen, 2008):

- It suffers from the state-space explosion problem, i.e., the number of states needed to model the system accurately may easily exceed the amount of available computer memory. Despite the development of several very effective methods to combat this problem, models of realistic systems may still be too large to fit in memory.
- Its usage requires some expertise in finding appropriate abstractions to obtain smaller system models and to state properties in the applied temporal logic.
- It is not guaranteed to yield correct results: as with any tool, a model checker may contain software defects.

2.5 Specification and verification in TLA+

A system's specification in TLA+ follows the *Standard Model*, which is the description of a set of behaviours each representing a possible execution of the system. Every TLA+ specification is composed of *predicates*, *actions* and *temporal formulas*.

Specifically, a *predicate* is a boolean-valued expression built from *variables* and *constants* and is evaluated on a state. The evaluation of a *predicate* in a *state* is performed by calculating the *predicate's* expression with the assigned *values* of the included *variables* in this *state*. A formal definition follows:

$$s[[p]] = p(\forall\, 'u' \colon s[[u]]/u) \tag{2.1}$$

where $p(\forall 'u' \colon s[[u]]/u)$ denotes the evaluation of predicate p by substituting every variable

u with the assigned value in state s ($s[[u]]$). If a *predicate p* is evaluated as *true* in *state s* then this states that *predicate p satisfies state s*.

An *action* denotes a relation between pairs of states of the system (denoted as system *steps*). Moreover, an *action* is a boolean-valued expression built from constants, *primed* and *unprimed* variables. *Primed* variables refer to new states while *unprimed* refer to old states. A formal definition of actions follows:

$$s[[a]]t = a(\forall\, 'u' \colon s[[u]]/u, t[[u]]/u') \tag{2.2}$$

We say that *action a satisfies states s,t*, or that s,t is an *a-step* if the evaluation of the action, with unprimed variables assigned value from state s and primed variables assigned value from state t, is true. A *predicate* can be considered as a special *action* that is evaluated only on the first state of a step.

A *temporal formula* in TLA+ is a boolean valued expression that is evaluated on behaviours. More specifically, a *temporal formula F* is composed by *action* and *predicates* combined with logical and temporal operators. An *action* / *predicate* can be considered as a special *temporal formula* that is evaluated on the first *step* / *state* of the behaviour. Some of the fundamental temporal operators in TLA+ follow:

- *Always* $\Box$. The formula F must satisfy every suffix of the behaviour. The semantics of the *always* temporal operator follows:

$$< s_0, s_1, \ldots > [[\Box F]] \triangleq \forall n \geq 0 \colon\ < s_n, s_{n+1}, \ldots > [[F]] \tag{2.3}$$

  where the $\triangleq$ operator utilized in an expression $id \triangleq exp$ defines id to be synonymous with the expression exp. Replacing id by exp does not change the meaning of the specification.

- *Eventually* $\Diamond$. The formula F must satisfy some states of the behaviour. The semantics or *eventually* can be defined by utilizing the *always* temporal operator as follows:

$$\Diamond F \triangleq \neg\Box\neg F \tag{2.4}$$

- *Leads to* $\rightsquigarrow$. The formula $F \rightsquigarrow G$ asserts that whenever F is true, G is eventually true. That is G is true at the same time with F or some time later. The *leads to* operator can be defined utilizing the previous temporal operators as:

$$F \rightsquigarrow G \triangleq \Box(F \implies \Diamond G) \tag{2.5}$$

Consequently, all the accepted behaviours of a system can be specified in TLA+ with a temporal formula called *specification* of the following form:

$$Spec \triangleq Init \wedge \Box Next \tag{2.6}$$

where Init is a predicate and Next is an action. A behaviour satisfies Spec if the first state of the behaviour satisfies Init and every consequent step satisfies Next. However, a well-defined specification should allow *stuttering* steps (steps that leave the system variables unchanged). Thus, the specification has the following form:

$$Spec \triangleq Init \wedge \Box[Next]_{<v1,\ldots,vn>} \tag{2.7}$$

where $[Next]_{<v_1,\ldots,v_n>}$ is defined as:

$$[Next]_{<v_1,\ldots,v_n>} \triangleq Next \vee ((v'_1 = v_1) \wedge \ldots \wedge (v'_n = v_n)) \tag{2.8}$$

The formula (2.7) demands that the transition from one state to another is either a Next-step or a stuttering step that leaves the system variables unchanged.

However, the TLA+ specification expressed in (2.7), also describes behaviours that may stop at any point, including a behaviour that starts in a valid initial state and takes no step. *Weak fairness* property of the action Next ($WF_{vars}(Next)$) asserts that behaviours are not allowed to stop in a state in which *Next* action is enabled (Action A is enabled in a state s if there exists a state t such that $s \rightarrow t$ is an Next-step). Therefore, the specification of a system that permits stuttering steps and supports the weak fairness property for action Next has the following form:

$$Spec \triangleq Init \wedge \Box[Next]_{<v_1,\ldots,v_n>} \wedge WF_{vars}(Next) \tag{2.9}$$

A verification property in TLA+ is described together with a system's specification and has the form of an *invariant* or a *property-temporal formula* (Lamport et al., 2002). An *invariant* is a predicate that is evaluated on a system state. Consequently, an *invariant* holds on a system specification, if and only if, every state of all the behaviours of a system satisfy that predicate. Moreover, a *temporal formula* is evaluated on a system behaviour. Consequently, a *temporal formula* hold on a specification, if and only if, it is satisfied by all the behaviors of the system.

2.6 Related work on the specification and verification of UCON

A number of formal models have been developed in order to capture the new semantics of continuity of decision and utilization of historical information in the usage decision making process through the attribute mutability mechanism. The formal model expressed by Zhang was developed in an extended version of TLA, developed also by Zhang, and mainly

targeted to highlight the enhanced expressiveness of UCON, compared to existing access control models (Zhang, 2006; Zhang et al., 2005). Additionally, the same specification sets as a level of atomicity the internal actions of the model and separates the actions that update the state of the usage from those that update the attribute value of the entities. However, the specification introduced in (Zhang et al., 2005) represents only a single usage process, and thus, it is raising questions on the operation of the usage control system where multiple usages are executing concurrently.

Zhang also analyzes the safety problem in a UCON system where new objects are created by proposing an additional model (Zhang, 2006; Zhang et al., 2006). The formal model expressed in (Zhang, 2006; Zhang et al., 2006) permits the existence of multiple usages. However it sets as level of atomicity the entire usage, thus eliminating the impact of concurrent operations into the internal state transitions of the usage states. Moreover, the formal model in (Zhang, 2006; Zhang et al., 2006) does not utilize a well-known formal language since it is described in a Z-like format. Both formal models proposed by Zhang are capable of expressing not only the model itself but also the policies supported by the model. This is done by specifying authorization predicates and attribute update procedures.

Another specification of a usage control was developed by Janicke in (Janicke et al., 2007) to support continuous on-going scenarios. Specifically, it is stated that within an initial usage entitled session, a number of "internal" usage requests may be performed. However, in order to support immediate revocation of usages, the latter is required to be performed by atomic actions. An alternative option states that the "internal" usage is not being revoked immediately but no additional usage requests are permitted. This specification was developed in ITL and it supports concurrent usages only within sessions. An additional formal model based on the POlicy Language based on Process algebra (POLPA) is presented in (Martinelli and Mori, 2010). This formalization presents only the basic features of UCON; does not deal with concurrency issues and does not present policy implementation

paradigms.

The verification of properties in usage control models has not been studied in (Janicke et al., 2007; Martinelli and Mori, 2010). The expressiveness and flexibility of the usage control model were presented in (Zhang et al., 2005) through the demonstration of policies such as RBAC, Chinese Wall, and so on. Moreover, the formal model in (Zhang et al., 2006) studied the safety problem in usage control concluding that only a very restricted pre-authorization subset of usage control models may support the safety problem. Work in (Ranise and Armando, 2012) also examined the safety problem in UCON policies. Nevertheless, it generalizes the restrictions imposed by (Zhang et al., 2006). Specifically, the formal model proposed in (Ranise and Armando, 2012) permits concurrent execution of usages and also allows attribute values to get values over infinitive domains using simple algebraic structure. A first attempt to formally analyze usage control policies with a model checker is presented in (Pretschner et al., 2009). Specifically, a formal model of simple usage control policies using an extension of Linear Temporal Logic (LTL). Thus, policies were analyzed by the NuSMV model checker tool, but no decidability result was derived. The authors in (Rajkumar et al., 2010) verified a policy implementation of a usage control system in SPIN. The implementation was designed for a Web based conference management application. However, it supports concurrent applications through a common communication channel. Furthermore, the usage scenario lacks of support for ongoing rules. Separation of duties policies in usage control is studied in (Lu et al., 2012), where initially they define a set-based specification of separation of duty policies and then they adapt it to attribute values of usage control. This results in applying already existing static mutually exclusive attribute (SMEA) constraints theory.

A comparison of the aforementioned information is depicted in Table 2.1

Model	Key Point	Ongoing	Concurrency	Level of Atomicity	Fairness	Strict to the Language	Policy
Zhang X. et al	Single Usage Process	Yes	No	Internal actions	No	TLA (No)	Yes
Zhang X. et al	Component Creation	Yes	No	Single usage	No	Z-like	Yes
Janicke H. et al	Series of usages within a session	Yes	Yes (session)	Internal actions	No	ITL (Yes)	No
Martinelli F. et al	Define activites during UCON process	Yes	No	Internal actions	No	POLPA (Yes)	Yes
UseCON	Concurrent usages, extended expresiveness	Yes	Yes	Internal actions	Yes	TLA+ (Yes)	Yes

Table 2.1 Comparison of related works

2.7 Chapter summary

In this chapter, we provided information about ODCE and related research work in the areas of access control models and their specification. Specifically, we highlighted the operational characteristics of ODCE along with the security threats presented in them. Moreover, we have given information on the RBAC, ABAC and UCON models, which are mostly applicable in the examined case of ODCE. Furthermore, we provided information on the specification of concurrent systems utilizing transition systems and verification utilizing the model checking technique. The process of system specification and verification in TLA+ was also presented next. The presented work on the specification and verification of models renders the aforementioned information valuable regarding the modeling of access or usage control systems since it can help in the identification of required specifications of access/usage control models for ODCE and also check the correctness of the designed or produced models in respect to their initial specifications.

Chapter 3

Requirements analysis

3.1 Introduction

Controlling the access to the resources of a system is an essential requirement for every computer security system (Lampson, 1974). Traditional access control models utilize only a single criterion for the allowance of an access request, which is related to the security characteristics of the subject and object involved in the requested access (Samarati and Vimercati, 2001a). More specifically, whenever a subject requests to access an object, the subject's clearance and the object's classification in Mandatory Access Control (MAC) models (Sandhu, 1993), the subject's identity in Discretionary Access Control (DAC) models (Qiu et al., 1985) and an activated role from a set of authorized to the subject in Role Based Access Control (RBAC) models (Sandhu et al., 1996), are being utilized accordingly. Attribute based access control approaches (OASIS, 2011) provide enhanced flexibility, when compared to the aforementioned access control models, by utilizing a number of subject and object security related characteristics, which are expressed in the form of attributes. However, recent research work highlights the fact that modern access control models, protocols and mechanisms are required to support modern computing environments. Challenges and

research directions in building models, protocols and architectures to support access control in pervasive/ubiquitous computing environments are presented in (Dritsas et al., 2006; Joshi et al., 2008; Thomas and Sandhu, 2004). A detailed categorization of access control requirements imposed by GRID and cloud systems are also presented in (Gouglidis and Mavridis, 2010; Theoharidou et al., 2013). Finally, the ASCAA (i.e., Abstraction, Separation, Containment, Automation and Accountability) principles for the next-generation RBAC model are presented in (Sandhu, 2008).

The Usage CONtrol (UCON) family of models (Park and Sandhu, 2004) provides an integration of traditional access control, digital rights and trust management. Moreover, UCON encompasses attribute-based characteristics, along with the concepts of continuity of decision and attribute mutability. Through the utilization of continuity of decision in UCON, access control to a resource is being controlled either continuously through an ongoing rule, or only before an access is permitted through a pre rule, as in traditional access control models. Therefore the term usage is preferred to be used instead of access. Moreover, the complexity of modern computing environments requires the utilization of a number of criteria during the usage control decision making process. UCON, employs three criteria for the creation of a usage decision, namely, the security related characteristics (henceforth called *properties*), contextual information and information regarding previous or current usages of the system's entities. However, whenever a subject s requests the usage of an object o, the usage control decision making can be based on either information related to s and/or o, or on information related to other system entities (e.g. father's *properties* may have an influence on the son's permissions), and henceforth mentioned as *direct* and *indirect* entities of the requested usage, respectively.

Despite the fact that the usage control decision making process in UCON utilizes all the three criteria, these are commonly related only to the *direct* entities. Additionally, the attribute mutability mechanism of UCON introduces a number of limitations regarding the

utilization of information about previous/current system usages. For example, no information about previous requested usages that were denied is recorded and no discrimination is done between the usages that have been revoked by the usage control system and the usages that have been terminated by a subject's request. Consequently, attribute mutability is unable to support a policy rule that is based on historical information regarding revoked usages. Moreover, modern computing environments present novel and complicated usage modes performed on objects by subjects, which are poorly supported through right entities in UCON. These complex operation modes require additional information that is essential for their execution, unlike the simple and straightforward operation modes that were previously supported by traditional access control models, e.g. read, write and execute operations in an operating system. For example, a banking transaction encompasses additional information, which is necessary for its operation, like the amount of transfer, the execution date etc.

In this chapter, in Section 3.2, we highlight the access control requirements imposed by ODCE. Furthermore, by recognizing the fact the UCON is the most promising access / usage control approach for ODCE, we continue in Section 3.3 with a detailed categorization of the usage decision criteria utilized in UCON along with representative usage scenarios.

3.2 Access control requirements

The following access control requirements have been resulted from an analysis of the ODCE characteristics through the application of different scenarios as presented in (Damiani et al., 2005; Grompanopoulos and Mavridis, 2010; Thomas and Sandhu, 2004).

Support of partially unknown-users: In several ODCE scenarios, the users of a system might not be known a priori by the system. Therefore, there is a requirement for an access control system that will be capable of making a decision based not only on the identity

of a user, but also on the user's security related properties (Damiani et al., 2005). In this case, the access control system gains in flexibility, as it is able to permit or deny access to partially unknown users. In order to satisfy the aforementioned requirements, the adoption of an attribute based access control approach is straightforward. Moreover, the utilization of historical information (i.e., transactions of a user in a system) in the decision making process helps in the creation of trust relationships, and thus, can be used for the definition of access control criteria (Artz and Gil, 2007; Boukerche and Ren, 2008; Kagal et al., 2001).

Cooperation among heterogeneous entities: A typical requirement in ODCE is the need for cooperation among different administrative authorities (Bacon et al., 2002). Therefore, in order to achieve collaboration among participants, it is necessary to have a common language regarding the definition of access control policies and entity attributes (Damiani et al., 2005). Moreover, the access decision creation in ODCE requires the utilization of information from a wide-range of participating entities. Specifically, whenever a subject s requests the usage of an object o, the usage control decision making can be based either on information related to s and/or o, or on information related to other system entities (e.g., father's properties may have an influence on son's permissions). We further on refer to them as direct and indirect entities of the requested usage, respectively. A basic design characteristic of an access control approach for ODCE should be its ability to utilize, in the decision making process, criteria that are related not only to *direct* but also to *indirect* entities.

Utilization of contextual information: The use of contextual information during the decision making process of an access control system consists of another requirement (Covington et al., 2001; Georgiadis et al., 2001; McDaniel, 2003). Thus, it is essential for a mechanism to collect (Baldauf et al., 2007) contextual information and evaluate it (Delir Haghighi et al., 2008; Duan and Canny, 2005; Ranganathan et al., 2004). Additionally, it should be possible to incorporate in access control policies context conditions as well as provide a unified model to represent contextual information (Cuppens and Cuppens-Boulahia, 2008).

Protection of privacy: A requirement of vital importance in ODCE is that of privacy (Cheng et al., 2005; Ranganathan, 2004; Theoharidou et al., 2013). For instance, the use of portable devices (e.g., mobile phones, tablets) by users, not only for business, but also for their everyday personal operations (e.g., contacts, photos) may raise privacy issues. Consequently, disclosure of personal information has a major impact on its users. Users require the full protection of their private information with minimal effort. Therefore, contextual information collected by sensors must also be protected as part of a user's private information (Minami and Kotz, 2005).

Ease of administration: The new computing paradigms that are realized in ODCE environments impose new requirements since they support a wide range of daily functions. These functions are performed by administrators who might have limited or no knowledge of the underneath technologies (Sandhu, 2008). Therefore, system features such as the support of complex access control policies are required to be performed transparently.

Resource constrained operations: There are several cases where a device in an ODCE might operate on batteries, and thus, has limited computing resources (Thomas and Sandhu, 2004). The constrained resources posed by ubiquitous environments prevent the use of highly secure and complex solutions (e.g., public-key infrastructure - PKI). Thus, in most cases it is required to use lightweight security approaches to ensure less power consumption and less usage of CPU and memory resources. Specifically, low bandwidth communication channels among participants require minimizing the communication overhead.

Adaptation on operational changes: A core feature of ODCE is that they operate in continually changing conditions. This is because, firstly, contextual information can be dynamic (e.g., time, available bandwidth), secondly, participating users can constantly change position, and thirdly, applications, data and devices may or may not be available for use (Kagal et al., 2003; Toninelli et al., 2007). Therefore, capabilities as decision mutability are required to be supported, especially in cases where usage control is applied. Addition-

ally, access control policies should be automatically altered as a result of environmental changes. In order to satisfy the aforementioned requirement, a continuous evaluation of the access control decision, (as it is expressed by usage control), should also be applied in the proposed model.

3.3 A critique on existing access control solutions

UCON is a next generation access control model capable of evaluating a number of usage decision criteria for the allowance or not of a usage request. Nevertheless, a limited utilization of the aforementioned criteria, related to the *indirect* entities, is being noticed. A detailed description of the criteria's utilization, along with corresponding representative usage scenarios[1], follows.

3.3.1 Security characteristics of entities

Security characteristics of system entities in UCON, are associated with subject and object attributes. These attributes are utilized by functional predicates (authorizations) that are evaluated for the usage decision. An example of a usage scenario, where only the subject's and object's *properties* are taken into consideration during the usage decision making process, is implementation of a MAC policy in UCON as presented in (Park and Sandhu, 2004). More specifically, in a system that implements a MAC policy the following rules apply:

Usage Scenario 1 *A clearance attribute is assigned to all the subjects of the system. Moreover, a classification attribute that shares the same value domain with clearance, is also assigned to all the objects of the system. A relation exists between the values of clearance and classification, thus creating a form of hierarchy. Consequently, a subject can read an*

[1] All the usage scenarios presented in this paper refer to pre authorization policy rules. However, the same criteria can also be applied to ongoing authorization rules.

object only if its clearance overcomes the object's classification. In addition, an object can be written by a subject only if its classification overcomes a subject's clearance.

Implementing the usage scenario 1 in UCON requires the utilization of authorization predicates that are evaluated on subject and object attributes. It is worth mentioning that UCON utilizes attributes for two purposes. More specifically, UCON does not only associates the entity's *properties* into the attribute values but also records into them the execution of system usages through the attribute mutability mechanism. Nevertheless, the values of the attributes that associate the *properties* of the entities are not updated automatically by the usage control system (update procedures) but only after the intervention of an administrator.

A limitation in UCON's authorizations is the fact that only the attributes from the *direct* entities are utilized. Nevertheless, in modern access control scenarios, it is possible that *properties* from *indirect* entities could also affect the authorization evaluation. A representative usage scenario that falls into the latter category is the following:

Usage Scenario 2 *Bob is a subscriber to a golf club that provides an amusement park for the children of its members. Bob's daughter, Alice is permitted to use all the available toys except from the carousel. Alice is permitted to use the carousel only if her father (Bob) is a member of the golden club category.*

When attempting to support the usage scenario 2 in UCON, Alice and carousel are considered to be the *direct* entities of the requested usage. However, during the usage decision making process, the values of Bob's attributes (e.g. his golden category membership) are also required. Due to the fact that Bob is an *indirect* entity, his attributes are not directly utilized in the corresponding authorization predicate. Nevertheless, if Alice is supported by an attribute "father" (that is assigned with the value of "Bob") then UCON is capable of resolving Bob's attribute values and consequently utilize them for the usage decision.

3.3.2 Contextual information

Contextual information in UCON is associated with special system variables, which are entitled condition variables. These variables are utilized in condition predicates in order to create a usage decision. A usage scenario, as originally presented in (Park and Sandhu, 2004), that requires the utilization of contextual information for the creation of the usage decision follows:

Usage Scenario 3 *The members of an institution are categorized into "faculty" and "student". The same categorization is also applied to the institution's areas. A member of a specific category (e.g. faculty) can exercise a right only in areas having the corresponding label (e.g. faculty areas).*

The presented approach in (Park and Sandhu, 2004), proposes the evaluation of condition predicates that contain condition variables, which associate the location information with *direct* entities. However, in case where contextual information that is associated with the *indirect* entities is required for the usage decision, UCON seems to be incapable of resolving which condition variable represents the contextual information that is related with a particular system entity. A usage scenario where contextual information, which is associated with the *indirect* entities, is utilized for the usage decision follows:

Usage Scenario 4 *The doctors in a hospital are categorized into "seniors" and "juniors" in respect of their operational experience. Every "junior" doctor is supervised by a corresponding "senior" doctor. Whenever a "junior" doctor, named Alice, sets a request for the execution of a specialized operation, e.g. an open heart surgery, a policy directive requires the physical coexistence of Alice's "senior" doctor supervisor, named Bob.*

A policy rule in UCON that models the usage scenario 4 requires the comparison between two locations represented by two separated condition variables. The problem arises

from the fact that Bob is an *indirect* entity. In such a case, specifying in UCON the particular condition variable that represents Bob's location seems to be impossible. In usage scenario 4, Alice is the *direct* entity of the usage request and only information related with her is utilized for the usage decision (condition). Even if UCON can represent with an Alice's attribute the fact that Bob is her supervisor, it is not possible to link at the same time Bob with a condition variable that represents his location. A solution could be provided by utilizing a number of condition variables that represent contextual information, which are irrelevant with the *direct* entities of the usage (e.g. "subjectsSupervisorLocation" may be the condition variable that represents the location of Bob). However, in a system with a large number of condition variables, such an implementation could result in a very complicated usage control system.

3.3.3 Historical information of usages

There are cases where historical information of previous or current usages executed by the *direct* entities, are needed to be utilized for usage decision. A usage scenario, where the previous usages of a subject may affect the allowance of a new usage requested by the same subject, is the following:

Usage Scenario 5 *An on-line collaborative educational software provides to its members the capability to post questions that can be answered by other members. However, a policy rule requires that a member is allowed to set a new question only if he had previously provided at least two answers to questions of other members.*

UCON is capable of supporting usage scenario 5 either through authorizations that incorporate attribute update procedures or through obligations. Specifically, if answering a question is considered to be a system usage, then each time a member provides an answer,

the value of an attribute that records this usage is being updated (attribute mutability). Consequently, an authorization predicate is evaluated, based on the value of the aforementioned attribute, to allow or not to the posting of a new question. More specifically, attribute mutability in UCON actually implements a mechanism that records the allowed system usages in attributes of *direct* entities. For example, every time a user listens to a music file, a specific attribute of her is being updated. The values of these attributes do not represent security characteristics of entities and are updated automatically by the attribute mutability mechanism. As a result, information about allowed usages is utilized for usage decision. However, attribute mutability faces a number of issues. Firstly, it provides limited knowledge regarding the system usages (only the allowed ones that contain attribute updates). Secondly, attribute mutability complicates the policy administration process by adding attribute update procedures to policy rules.

Giving an answer to the question in usage scenario 5, can be considered as an obligation operation that must be executed twice as a criterion for allowing a usage request to post a new question. Obligation operations in UCON also represent usages that are exercised by subjects on objects. However, these obligation operations are discriminated from normal system usages because they are not controlled by a decision factor (Authorization, oBligation or Condition) and can be performed whenever required (Zhang et al., 2005). Nevertheless, in modern computing environments, it is possible for the usage decision to be dependent on past usages of *indirect* entities. A usage scenario that falls into this category is the following:

Usage Scenario 6 *In a research institute, a presentation room is equipped with both an interactive board and a media player. A policy rule requires that an employee is permitted to access the media player only if there is no other presentation in progress (usage of the interactive board) in the same room.*

Usage scenario 6 can be modeled only through UCON's obligations and not through authorizations that incorporate attribute mutability update procedures (as happened with usage scenario 5). Authorizations with attribute mutability fail to model scenario 6 because only the attributes of the *direct* entities of a usage are being updated. Moreover, authorizations utilize only attribute values from *direct* entities. Thus, the usage of the media player in usage scenario 6 without the utilization of obligations, seems to be impossible. However, a significant drawback of obligations is the lack of a feasible fulfillment mechanism, as it is mentioned in (Lazouski et al., 2010).

Therefore, we summarize the utilization of UCON's usage decision criteria in Table 1, based on the analysis performed in the aforementioned scenarios. The usage decision criteria are represented as rows on the left side of the table. These are, as identified, the *properties*, *context* (contextual information), and *history* (information regarding previous or concurrent usages) of the system entities. The far right two columns of the table represent the origin of the aforementioned criteria, which can stem from either a *direct* or an *indirect* entity. Thus, each usage decision criterion, originating from an entity, is utilized by UCON's decision factors that are expressed in the corresponding cell. Each UCON decision factor is represented by a letter (Authorization, oBligation, Condition) combined, if required, with the attribute mutability mechanism. For instance, if a usage decision criterion is based on historical information stemmed from *direct* entities, then UCON is capable of utilizing it by using either authorizations with attribute mutability (A+m) or obligations (B).

Supporting Complicated Usage Modes

Rights in UCON are described as "privileges that a subject can hold and exercise on an object" (Park and Sandhu, 2004). However, rights are not described by attributes, as opposed to subjects and objects. Such a modeling decision seems to be adequate for simple and straight-forward rights, like reading or writing a file. However, in modern computing envi-

Table 3.1 Utilization of decision criteria in UCON

	Direct entities	**Indirect** entities
Properties	A	A
Context	C	-
History	A+m, B	B

ronments subjects may need to access objects with novel and complicated access modes. A usage scenario that requires a complicated usage mode is the following:

Usage Scenario 7 *Electronic banking transactions must confirm with a government policy directive requesting that players of on-line betting companies must be adults. More specifically, whenever a customer attempts to make a money transfer to a betting company's account, his age must be evaluated. In addition, the amount of money transfer must not overcome the customer's account balance.*

In the usage scenario 2, the customer's account is the subject of the usage, whereas the money transfer is the right and the account of the betting company is the object. The necessity for associating the amount of money with the money transfer results in the need for enhancing rights with a more detailed description.

A possible solution in UCON, to overcome the lack of right attributes, might include decomposing the transaction of usage scenario 2 into two different rights. In the first right, the customer's account is the subject, whereas the money transfer is the object. In the second, subject is the money transfer and object is the bank account of the betting company. For each one of these rights, a respective UCON policy rule is created. However, utilizing these two UCON policy rules has significant drawbacks. Firstly, UCON modeling does not depict that these two policy rules are related with each other. Only the policy administrator is aware that these two rules are correlated and must be checked in a particular order.

Secondly, and more importantly, is the fact that UCON's modeling is quite different from the way security policies are expressed, usually through natural languages, e.g. the same way assembly differs from SQL programming. As a result, UCON introduces additional complexity to the policy administrator tasks. Thirdly, the fact of using only simple rights (e.g. read, write, etc.) limits the policy administrator's ability to create a rule that controls either all the rights or only a specific one.

The replacement of UCON's *right* component with a new one, called *action*, could provide a solution to the previous described UCON right shortcomings. Actions can be described, like subjects and objects, with attributes. It is worth mentioning that the necessity for further enrichment of UCON with right attributes was also proposed in (Zhang and Parashar, 2004). Moreover, actions can support efficiently complicated rights. The previous example with the banking transaction can now be modeled by utilizing an action having attributes that describe the transaction amount, the purpose of money transfer, etc. Actions, along with their attributes, can contribute to the increasing of policy language expressiveness. In addition, action attributes can reduce the gap between UCON modeling with multiple policy rules and expressing security policies in almost natural language. Action attributes can also enable the creation and management of possible existing right hierarchies, as presented in (Park and Sandhu, 2004).

Moreover, a detailed analysis of the challenging issues that UCON presents in ODCE are depicted in Table 3.2.

3.4 Chapter summary

In this chapter, we highlighted through representative usage scenarios and related literature the additional requirements that are posed when attempting to implement the access control process to ODCE. Moreover, a critique on existent access control approaches, focusing on

Table 3.2 An analysis of challenging issues of UCON in ODCE

Usage Decision Criteria	UCON mechanism	UCON submodel	Comments
Security Characteristics of Entities	Subject and Object attributes	**A**	Attribute values are updated through an administrative process
Contextual Information	Condition Variables	**C**	Conditions may utilize contextual infromation only directly related with the entities of the usage requested
Historical Information of Usages	Attribute mutability	**A**	Complicated policy administration process Limited Knowledge of previous usages
	Obligations management	**B**	Absence of an obligation fulfillment enforcement mechanism "Obscure" discrimination between obligation operations and normal usages

UCON, is also presented. Specifically, a classification of usage decision criteria, originating from either *direct* or *indirect* entities, highlighted the limitations of the UCON model and spotted out the necessity for a new use-based usage control model.

Chapter 4

The proposed model

4.1 Introduction

This chapter presents our proposed UseCON model for ODCE, which is designed in regard to the requirements identified in Chapter 3. The remainder of the chapter is organized as follows. Section 4.2 provides information regarding the proposed model including its main entities and relations between them. In Section 4.3, we provide a formal specification of our proposed model in TLA+, and in Section 4.4, we illustrate a policy specification also in TLA+ through an example. Section 4.5 concludes the chapter.

4.2 A brief description

The UseCON model is composed by three entities, namely subjects, objects and actions. These three entities, together with uses, are the core components of UseCON. Decision factors in UseCON are the attribute dependent authorizations and the usage dependent authorizations.

Subject and object are fundamental concepts, proposed already by primary access con-

trol models. More precisely, a subject is an entity that requests the execution of an operation on another entity named object. An action entity represents the novel and complicated operations of subjects on objects, imposed by modern computing environments. Moreover, all the security relevant characteristics, including related contextual information, of subjects, objects and actions are described through their attributes. An example of an action entity is a money transfer of a bank account with attributes describing the amount of transfer, the date of execution, the currency etc. A core component of the UseCON model is the use component which represents the security related semantics of a usage. A use, is created when a subject requests the execution of an action on an object. A use is described through attributes that record the detailed security-relevant characteristics and capabilities that are associated with the requested usage. Each use is further associated with a state attribute, which embodies the accomplished status of the usage in progress, as it is described in (Zhang et al., 2005) and it is depicted in Figure 4.1. The state attribute receives each time one of the following values (Grompanopoulos et al., 2013):

- Requested: Upon request for a usage, the appropriate attributes are associated with the use and proper values are assigned to them. The pre-authorization policy rules, which govern the requested usage, has not been evaluated yet.
- Activated: The requested usage has been allowed, as a result of successfully fulfilled pre-authorization policy rules, and is being executed.
- Denied: The requested usage has been denied, because it failed to satisfy the pre-authorization rules.
- Stopped: The allowed / ongoing usage has been terminated by the system due to a violation of an ongoing authorization rule.
- Completed: The usage that has been completed due to a subject's intervention.

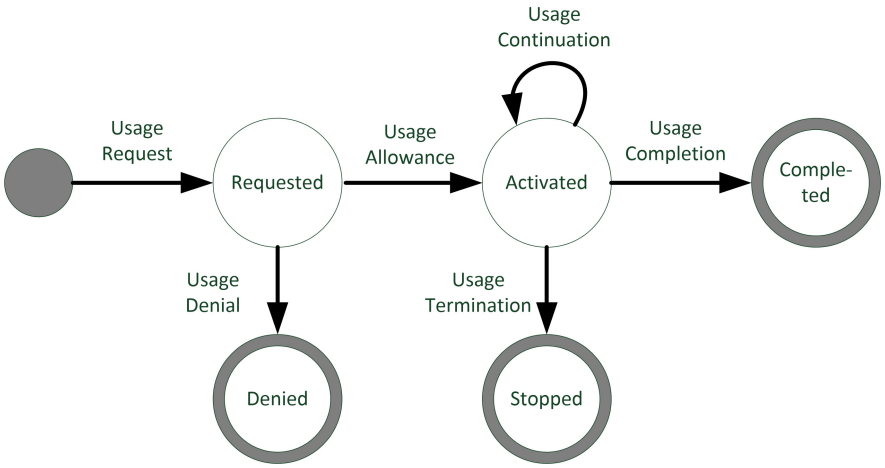

Fig. 4.1 Accomplishment status of a single usage

An authorization is the only decision factor in UseCON. However, for the creation of a usage decision, the UseCON model utilizes three criteria viz. the (*properties*) of the entities, contextual information and historical information about usages. Therefore, authorizations are categorized into Attribute dependent Authorizations (AdAs) and Usage dependent Authorizations (UdAs) as follows:

- Contextual information and *properties* that describes an entity are associated with the corresponding entity's attributes. The values of these attribute in turn are utilized by AdAs policy rules for the creation of a usage decision.

- Historical information of usages is utilized by UdAs. Specifically, in UseCON, uses record all the information regarding the previous or concurrent usages exercised in the system. Consequently, a UdA policy rule utilizes the historical information contained into the use attribute values to allow or deny a usage request.

Integrating authorizations with continuity of decision results into two UseCON sub-models. These are the pre-Authorizations and the ongoing Authorizations sub-models. The UseCON elements and the relations between them are depicted in Figure 4.2.

UseCON presents extended expressiveness, as required by modern computing environ-

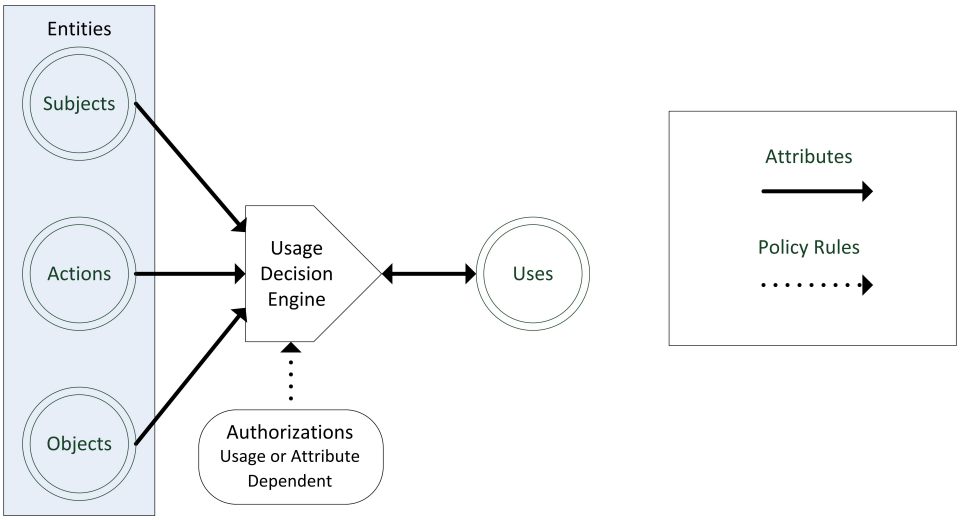

Fig. 4.2 UseCON usage control system

ments, not only due to the fact that is able to utilize all the three criteria (i.e., contextual information, *properties* and historical information) but also because these criteria can be related to either *direct* or *indirect* entities or even to any subset of the usage control system entities, e.g. a bank should issue new loans to a customer only if the sum of the existing loans of all customers is lower than a given amount. Moreover, UseCON inherently supports the utilization of historical information of usages and not through the application of attribute update mechanism as UCON does. Consequently, there is a strict distinction between the functional components of the internal usage control model (e.g., creation and state transition actions of use entities) and the components that define the specific policy implementations of the model (e.g., the creation of the usage decision – policy rules).

4.3 Specification of UseCON in TLA+

In this section, we present a logic based approach to formally define UseCON in TLA+. Specifically, basic elements of specifications are presented together with the representation of policy rules and use attribute update procedures. Moreover, the transition system of both

a pre and an ongoing usage control system that supports the concurrent operation of multiple usages, are followed.

4.3.1 Basic elements

A system's usage represents the request for the execution of an action (a) from a subject (s) on an object (o). All subjects, objects and actions utilized in the usage control system define the sets of subjects (S), objects (O) and actions (A), respectively. Set E refers to the union of S, O and A sets, as follows:

$$E \triangleq S \cup O \cup A$$

The security related characteristics of an entity are represented through its attribute values. An attribute is a function whose domain is a particular set (S or O or A) and its range is composed of specific attribute values, as follows:

$$Att_i(e) \in [e \in E \mapsto RangeAtt_i]$$

For every system entity an identification attribute *id* is defined for assigning a unique value to the entity that remains constant through the life-time of the usage control system. Thus, the following invariant is valid for all the behaviours of the usage control system.

$$\forall e_1, e_2 \in E : id[e_1] = id[e_2] \Rightarrow e_1 = e_2$$

In UseCON, the operation of the usage control system does not modify automatically the attribute values of system entities (S, O or A). Thus, the manipulation of attribute values of system entities is proposed to be covered manually by the usage control administration model. Consequently, a system entity is represented with a constant record having as fields the entity's attribute values, as follows:

$$e \triangleq [id \mapsto k, att_1 \mapsto l_1, att_2 \mapsto l_2, \ldots, att_n \mapsto l_n]$$

where $e \in E$ is a system entity and $att_i, i \in 1, \ldots, n$ is a value of one of its attributes. The notation $e.att_i$ represents the value of the attribute att_i of entity e. The first record field of every entity is the *id* attribute. Therefore, the set of subjects, objects and actions (S, O and A respectively) are defined, as follows:

$$S \triangleq \{s_1, \ldots, s_n\}$$

$$O \triangleq \{o_1, \ldots, o_n\}$$

$$A \triangleq \{a_1, \ldots, a_n\}$$

A usage request in UseCON results into the creation of a use. A use materializes the accomplishment status of the usage in progress and it is described with attributes. More specifically, every use instance must contain an use's id attribute with value a tuple composed of the attributes values *sid*, *oid*, *aid* that describe the identities of the subject, the object and the action participating in the materialized usage. A special attribute *st* is associated to every use instance representing the accomplishment status of the usage, as presented in subsection 4.2 taking one of the values requested, activated, denied, stopped and completed.

A use instance u that materializes a specific usage request from subject s to object o for action a is represented in the specification of the model with a variable record, having as

fields its attribute values as follows:

$$u \triangleq [sid \mapsto s.id, oid \mapsto o.id, aid \mapsto a.id, st \mapsto state,$$
$$uatt_1 \mapsto v_1, uatt_2 \mapsto v_2, \ldots, uatt_n \mapsto v_n]$$

where s.id, o.id and a.id are the identity values of the subject, the object, and the action, respectively. The state attribute *st* gets a value *state* which belongs to the following set: $state \in \{$"requested", "activated", "denied", "completed", "stopped"$\}$, $u.att_i, i = 1,2,\ldots,n$ are the use attributes. The set of all the system uses is U. During the operation of the usage control system, U is populated due to the usage requests. Moreover, as these usage requests are served by the usage control system, the use attribute values are modified. Specifically, the U is altered whenever a new usage is requested or a usage changes its progress status (e.g., from *activated* to *stopped*) through the accomplishment of a system action, as are described in the next subsection. Consequently, the only variable utilized in the specification is U, which is the set containing all the usages operated in the UseCON model and it is declared as follows:

$$VARIABLES \quad U \tag{4.1}$$

Initially, for every TLA+ specification, particular modules are included that permit the utilization of specific operators. In our specification, the *finiteSets* and *Integers* modules are included that encompass arithmetic and set - related operators, like *Cardinality*. The semantics of module declaration are the following:

$$EXTENDS\, Integers, FiniteSets$$

Decision Making in UseCON

The flexibility of UseCON's policy rules to utilize information originated from a multitude of sources, contributes to the extended expressiveness of the model. For example, authorization predicates in UCON are categorized into unary predicates, which compare an attribute value with a constant value, and binary predicates, which compare two attribute values (Zhang and Parashar, 2004). However, both unary and binary predicates utilize attribute values related only with *direct* entities. Nevertheless, policy rules in UseCON provide an enhanced utilization of information from entity and use attribute values. More precisely, the general form of a UseCON policy rule that governs the allowance of a usage request from a subject s on an object o with an action a, is a boolean valued expression with semantics as follows:

$$Policy\_Rule(s,o,a,S,O,A) \triangleq expression(e_1,\ldots,e_n)$$

where s,o,a are the particular direct entities of the usage and S,O,A are the sets of all entities. In addition, two or more UseCON policy rules can be combined together with logical operators as follows:

$$p = p_1 \otimes p_2 \otimes \ldots \otimes p_n$$

where $\otimes$ is a logical operator, (e.g. AND, OR, etc), and $p_i\, i = 1,\ldots,n$ is a policy rule.

The parameters $e_i : i \in 1,\ldots,n$ of a policy rule that are utilized for the evaluation of the *expression* may have various origins, and thus lead to the creation of the following categories of policy rules:

- *Direct Policy Rules*: The parameters e_i in the expression of a direct policy rule are values only from attributes of *direct* entities or constant values. Specifically, all pa-

rameters e_i, are defined by the following formula:

$$e_i \in \{s, o, a\} \; or \; e_i \triangleq l$$

where l is a constant value. Examples of using direct policy rules of entities might be the expression which verifies that the age of a subject is over 18. Another direct policy rule is the expression that evaluates if the clearance of a subject is greater than the classification of an object. In both aforementioned examples security properties related only with the *direct* entities, i.e. age of subject, clearance or classification of subject and object, are utilized in the policy rule.

- *Indirect Policy Rules*: The expression of an indirect policy rule consists of attribute values stemmed not only from direct, but also from indirect entities. However, there is a logic relation between the two types of entities (i.e., direct and indirect) which is represented in some attribute values of the entities. For example, an employee named Alice should have an attribute "supervisor" assigned with the value of the id attribute of Alice's supervisor. Consequently, the utilization of the *indirect* entities in the policy rule is possible, through an appropriate expression (i.e., *selection*) that takes parameters the attribute values of direct entities. The semantics for the *selection* expression follows:

$$e_i \triangleq CHOOSE\, x \in E : select(x, s, o, a, l)$$

An example of an indirect policy rule is an expression which evaluates if the father of a child is a member of a "golden" category class. In this example, the child is a *direct* entity where the father is an *indirect*. The *selection* expression should utilize the fact that there is a "father" attribute in the child entity having the attribute value of

her father's identity.

- *Complex Indirect Policy Rules* Modern computing environments impose complicated access control policies where the usage decision is based on information related not only with a single entity, but with a subset of entities. Such complicated policies can be supported in UseCON through complex indirect policy rules. More specifically, a parameter e_i of a complex indirect policy rule can be, apart from a single (direct, or indirect) attribute value, an aggregation of information. This information is derived from all the entities that satisfy a desired (*select*) predicate. The semantics of a parameter e_i of a complex indirect policy rule follows:

$$e_i \triangleq aggregation(\{e \in E : select(e)\})$$

An example of a complex indirect policy rule is one that confirms that the sum of the balances from all the accounts of a bank's customer is over a specific amount. Information that is related with a set of bank accounts, those belonging to the corresponding user, is required for the evaluation of the aforementioned policy rule. Consequently, the *selection* expression defines the subset of the bank accounts that belongs to the specific customer.

Use Attribute Update Procedures

Attribute values are utilized by UseCON policy rules through the usage decision making process. Consequently, the implementation of a high level policy should also cope with the definition of use attribute value update procedures because these values determine the outcome of the policy rule. However, attribute mutability of uses in UseCON has a different objective from that by UCON (Park and Sandhu, 2004). Specifically, UseCON attribute mutation does not support historical information, e.g., how many times an object has been

accessed, which is now supported by the creation of use entities. Instead the mutation of attribute values only records security related information that are related with the usage. Moreover, a categorization of the use attributes according to the nature of information they record follows:

- *Induced Attributes*. Information that can be inferred from entities involved in a usage (i.e., subject, object and action) that a specific use materializes. An example of *induced* use attribute is the price of a service.

- *Observed Attributes*. Information recorded during the exercise of a usage. Such info cannot be derived directly from the entities involved in the usage. Examples of *observed* use attributes is the duration of the usage, or the set-time of usage request etc.

Update procedures for *induced* attributes can be specified during the definition of the implementation of a high level policy by the policy administrator. However, update procedures for *observed* attributes require the existence of a system function that returns the required value. For example, a policy's specification, which requires to record the system time whenever a usage is permitted, follows:

$$preUpdate \triangleq [u\ EXCEPT\ !.st = "activated",\ !.allowedtime = SystemTime()]$$

where *SystemTime()* is an internal function provided by the system framework that provides the current time.

Attribute update procedures in UseCON are performed whenever a usage changes its state (e.g., from "requesting" to "activated"). Therefore, during the execution of a usage, the times of use attribute update in UseCON are represented in Figure 4.3.

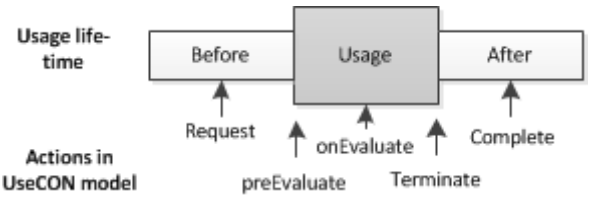

Fig. 4.3 Use attributes updates during the exercise of a usage in UseCON

4.3.2 Transition systems

Actions in the UseCON model are categorized to those triggered by a subject's request and those operated automatically by the usage control system. More specifically, for every usage supervised by the usage control system the following actions can be triggered by a subject (Figure 4.4 and 4.5):

- *Request*: This action performs the transition from the "Init" state of the usage to "Requested". Moreover, *Request* creates a particular use instance that materializes the requested usage and also assigns values to the use's id attribute.
- *Complete*: This action changes the usage's state from "Activated" to "Completed".

The actions performed automatically by the usage control system, follows:

- *preEvaluate*: This action is performed by the usage control system only when the usage's allowance is governed by a pre-authorization rule. This action changes the usage's state to either "Activated" or "Denied", depending on the outcome of the examined policy rule.
- *onEvaluate*: In case the allowance of a usage is governed by an ongoing authorization rule, the *onEvaluate* action is performed by the usage control system. If the particular policy rule is satisfied then the usage's state remains unchanged. In case the policy rule is not satisfied, the usage's state is changed to "Stopped".

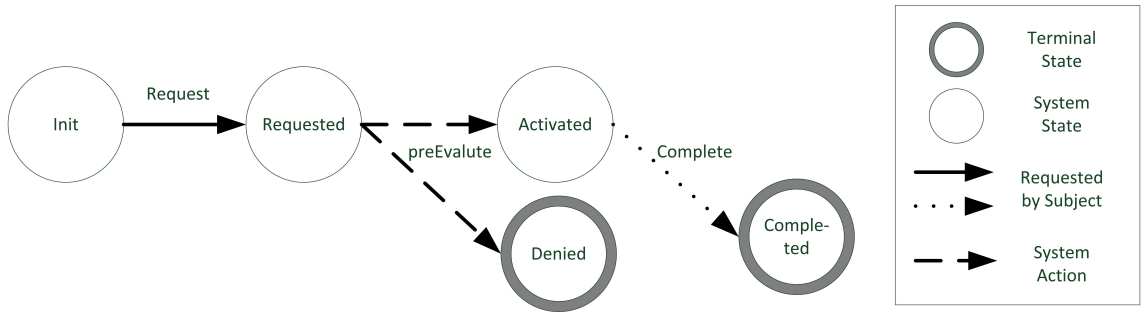

Fig. 4.4 Transition system of a single usage pre-authorization UseCON model

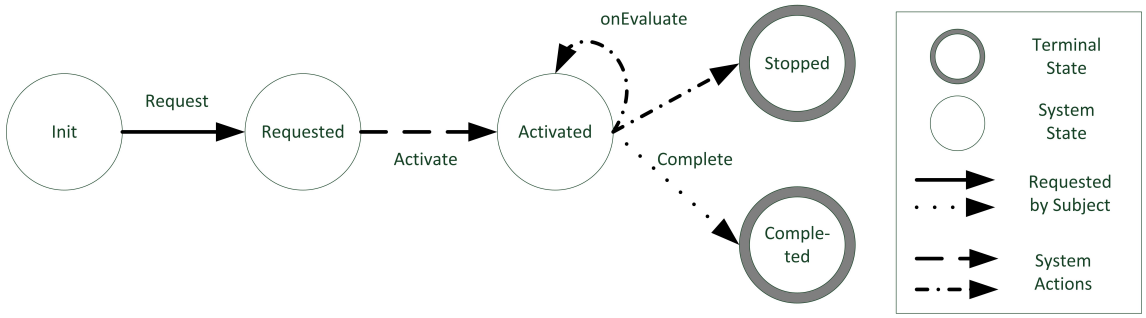

Fig. 4.5 Transition system of a single usage ongoing authorization UseCON model

- *Activate*: This action is performed only when the usage's allowance is governed by an ongoing authorization rule. It follows the execution of the *Request* action and changes the usage's state from "requested" to "activated".

Any of the previous actions apart from modifying the *st* attribute value they may also update other use attribute values. Such modifications in the use attribute values are considered to be implementation specific, as already described in subsection 4.3.1. Moreover, despite the existence of a great number of updates on attribute values, the execution of any of the previous actions is considered to be atomic [1], i.e. a single step of a behaviour.

The transition system for a single usage UseCON system controlled by a pre and an ongoing authorization policy rule is depicted in Figure 4.4 and Figure 4.5 respectively.

In initial state of the UseCON's transition system where multiple usages are operating concurrently no usages are exercised in the system, and thus, the first state of every be-

[1] This constraint is realistic due to the fact that a use is recorded centrally on the policy decision point and there is no need for attribute updates of other entities (subject or object)

haviour must satisfy the TLA+ predicate *Init* which defines that the set-variable U is an empty set:

$$Init \triangleq U = \{\}$$

Moreover, according to the time period that a usage request evaluation is performed, a Pre-Authorization and Ongoing Authorization transition systems are created. The TLA+ specification follows:

Pre-Authorization

As it is depicted in Figure 4.4 the possible actions that can be performed on a pre-Authorization UseCON system are either the usage request for a new usage, either the evaluation of an already requested usage or the termination of a usage that is already executed. Therefore, the *Next* action, that describes all the possible next states could be either a *Request* or a *Evaluate* or a *Complete* action and is described as follows:

$$Next \triangleq Request \vee preEvaluate \vee Complete$$

More specifically, the *Request* action selects non- deterministically [2] a new usage x. This is verified by searching the set of uses U, and returning one that has not already been requested by the subjects or processed by the usage control system. Consequently, in case that this particular x exists, the *request* action creates the corresponding use instance that materializes usage x and inserts it to the set U. The semantics of *Request* action follows:

[2] the non-determinism property is implied by the use of $\exists$ operator. For comprehensive information refer to (Lamport, 2002)

$$Request \triangleq \exists u \in (S \times A \times O) : ($$
$$\wedge \forall x \in U : (x.sid \neq u[1].id \vee x.aid \neq u[2].id \vee x.oid \neq u[3].id)$$
$$\wedge U' = U \cup \{createUse(u)\})$$

The *preEvaluate* action examines if there are any usages that have been requested but have not been processed by the usage control system. More specifically, *preEvaluate* examines if there is any use instance with state attribute value equals to "requested". Consequently, the action evaluates the policy rule that governs the allowance of the usage that the specific use instance materializes. Based on the outcome of that policy rule the action modifies the state of the use either to "activated" or to "denied" with *preUpdate* and *denUpdate* use attribute update procedures respectively as follows:

$$preEvaluate \triangleq \exists u \in U : (\wedge\ u.state = \text{"}requested\text{"}$$
$$\wedge IF(PolicyRule)\ THEN$$
$$U' = (U \setminus \{u\}) \cup \{preUpdate(u)\}$$
$$ELSE$$
$$U' = (U \setminus \{u\}) \cup \{denUpdate(u)\})$$

Action *Complete* simulates a subject's request to terminate the execution of a currently active usage. If such a use exists, its state attribute value should be equal to "activated". Consequently, the *completed* action modifies the state attribute value from "activated" to "completed" with the *comUpdate* use attribute update procedure. The semantics of the

Complete action follows:

$$Complete \triangleq \exists u \in U : (\wedge\, u.state = \text{"}activated\text{"} \\ \wedge\, U' = (U \setminus \{u\}) \cup \{comUpdate(u)\})$$

CreateUse is the procedure that creates a use instance that has the identities of the *direct* subject, object, action. The semantics of the *createUse* procedure follows:

$$createUse(x) \triangleq [sid \mapsto x[1].id, aid \mapsto x[2].id, oid \mapsto x[3].id, \\ state \mapsto requested, att \mapsto k]$$

All the other use attribute update procedures perform a dual role. Firstly, they alter the *state* use attribute to the desired value (e.g. *preUpdate* to "activated" or *comUpdate* to "completed"). Secondly, the use attribute update procedures modify the values of other use attributes according to the requirements of the usage control system that they are called to describe. For example, a possible *preUpdate* procedure can be the following:

$$preUpdate \triangleq [u\ EXCEPT\ !.st = \text{"}activated\text{"},\ !.att = value]$$

where the *EXCEPT* in TLA+ is a special purpose operator representing the modification of a function from a state to the next. In that next state all function values are left unchanged unless stated otherwise [3]. The same semantics apply to the *denUpdate, comUpdate* procedures.

The transition system of a pre-Authorization UseCON system with two usages that are operating concurrently is depicted in fig 4.6. For figure simplicity reasons it is assumed that the policy rule is always valid so every usage that is requested is consequently activated.

[3] For a comprehensive definition of EXCEPT operator refer to (Lamport, 2002)

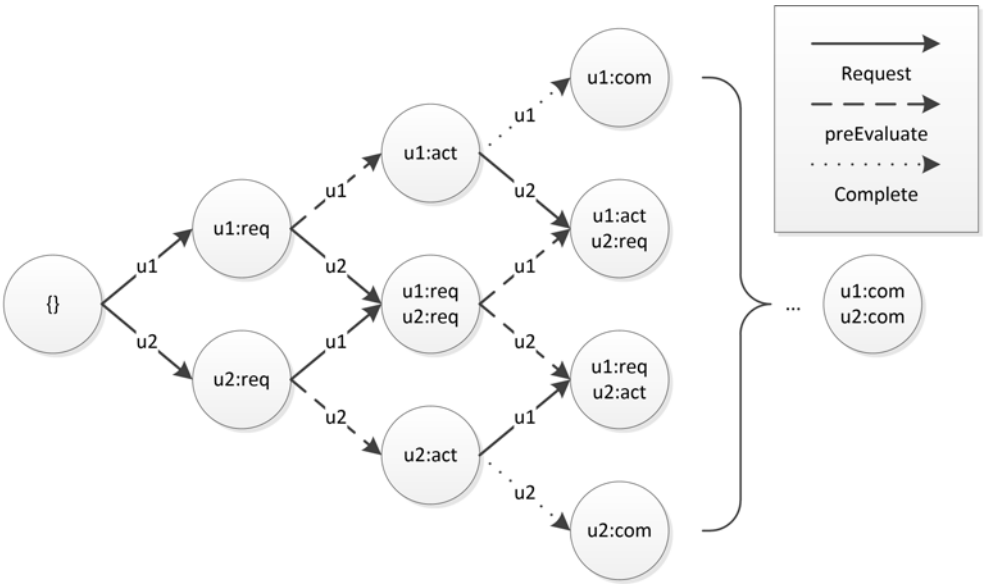

Fig. 4.6 Transition system of a multiple usage pre-authorization UseCON model

Ongoing Authorization

The transition system of the Ongoing Authorization UseCON model is differentiated from the pre Authorization model in a number of ways (as it is depicted in differences between Figure 4.4 and Figure 4.5). Firstly, in an ongoing model, a usage that is requested is permitted to be activated without the evaluation of any policy rule. Secondly, at a given time interval [4], an ongoing action *onEvaluate* is executed. Thus, the specification of the *Next* action on an ongoing authorization model has the following semantics:

$$Next \triangleq Request \vee Activate \vee onEvaluate \vee Complete$$

The *Activate* action searches for the existence of a use with state attribute value equal to"requested" and consequently updates it to "activated" by executing the *preUpdate* proce-

[4]The determination of the exact interval is left open as an implementation issue

dure.

$$Activate \triangleq \exists u \in U : (\wedge\ u.state = \text{"}requested\text{"} \\ \wedge U' = (\{U \setminus \{u\}) \cup \{preUpdate(u)\})$$

Whereas *onEvaluate* action evaluates an ongoing policy rule and based on this result it either leaves use to "activate" state, but it can possibly update the rest use attribute values with *onUpdate* procedure, or modify its state attribute value to "stopped" with *termUpdate* use attribute update procedure as follows:

$$onEvaluate \triangleq \exists u \in U : (\wedge\ u.state = \text{"}activated\text{"} \\ \wedge IF(PolicyRule)\ THEN \\ U' = (\{U \setminus \{u\}) \cup \{onUpdate(u)\} \\ ELSE \\ U' = (\{U \setminus \{u\}) \cup \{stopUpdate(u)\})$$

Moreover, in the case where there is no need for an ongoing use attribute update procedure, the *onUpdate* procedure can be substituted with the *UNCHANGED U* TLA+ operator that leaves the variable U unmodified.

The semantics of *Request* and *Complete* actions are the same with the pre-authorization model. Moreover, the transition system of an ongoing UseCON system with two usages that are operating concurrently is depicted in Figure 4.6. For figure simplicity reasons it is assumed that the policy rule is always valid so there is no usage that is terminated by the system.

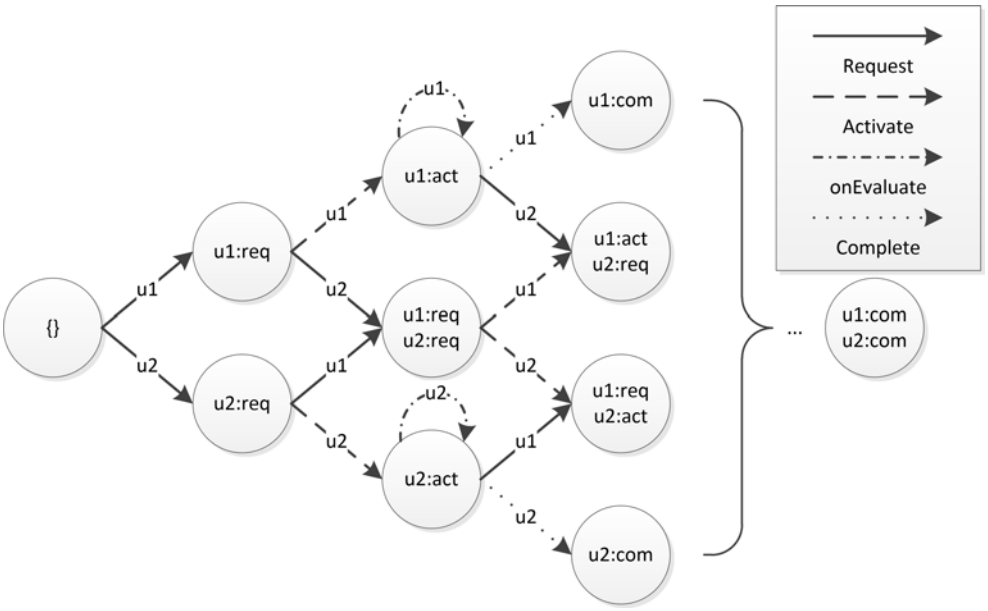

Fig. 4.7 Transition system of a multiple usage ongoing authorization UseCON model

4.4 Example of a policy specification

The development of an access/usage control system is a multi-layer process that results in the definition of an access/usage control policy, model and mechanism (Samarati and Vimercati, 2001a). A policy declares the high level directives that regulate access to resources, while the model is a formal representation of the system. Moreover, the mechanism defines the low-level hardware and software functions that implement the desired policy. UseCON is a general purpose usage control model that is capable of supporting a wide range of high level policies (Grompanopoulos et al., 2013). However, for the implementation of a specific high-level policy through UseCON, the specification of policy rules and use attribute value update procedures is required. Policy rules are required because they are responsible for getting a decision regarding a usage allowance. Additionally, attribute value update procedures indirectly determine a usage decision, and consequently the policy. This is feasible by mutating the attribute values utilized by policy rules. A motivating scenario that presents a high level usage control policy follows:

Usage Scenario 8 *An educational institute provides to its members a number of computational resources to utilize. A resource's usage can be categorized to either academic or personal. In order to provide the desired Quality of Service (QoS), the institute defines a high level policy stating that an upper bound applies on the number of simultaneous usages per resource.*

A number of usage revoking strategies can be implemented, when the upper limit of simultaneous usage on a resource is reached. For instance, a possible strategy could state that "academic" oriented usages should override "personal" usages on the same resource. Another strategy, motivated by "social fairness", could require revoking the usage requested by the subject that occupies the greatest number of resources in the system.

The implementation of the aforementioned strategies, using a pre-authorization UseCON model, requires that whenever a usage on a resource is requested, it must be checked if the total number of usages performed on this resource has reached out to a maximum number. Therefore, if the number of usages reaches out the maximum number then a revoke-usage strategy is applied. This, results in revoking either the requested or the already activated usages. If not, the requested usage is promptly activated. A specification in TLA+ that implements the aforementioned operations follows:

$$
\begin{aligned}
preEvaluate \triangleq \quad & \exists u \in U : (\\
& \wedge u.state = \text{"}requested\text{"} \\
& \wedge \quad \vee \quad (\wedge NumOfUsonObject(u) \neq MAXLIMIT - 1 \\
& \qquad \wedge U' = (U \setminus \{u\}) \cup \{preUpdate(u)\}) \\
& \qquad \vee \quad (\wedge NumOfUsonObject(u) = MAXLIMIT - 1 \\
& \qquad \wedge LET\ act \triangleq CHOOSE\ x \in U : SameObjectAct(x,u)\ IN \\
& \qquad IF \quad (Policy\_Rule(act,u)) \\
& \qquad THEN \\
& \qquad U' = (U \setminus \{u\}) \cup \{termUpdate(u)\} \\
& \qquad ELSE \\
& \qquad U' = (U \setminus \{u\}) \cup \{preUpdate(u), termUpdate(act)\})
\end{aligned}
$$

where *NumOfUsonObejct* returns the number of concurrent usages on the resource that usage u requests and it is defined as:

$$NumOfUsonObject(x) \triangleq Cardinality(u \in U : x.oid = u.oid)$$

and *SameObjectAct* declares the set of activated usages [5] on the same resource with the requested usage and is defined as:

$$SameObjectAct(x,y) \triangleq x.oid = y.oid \wedge x.st = \text{"}activated\text{"}$$

[5] For reasons of simplicity and limitations in the software that performs verification of TLA+ specifications, we use a single entity set instead of a *SameObjectAct*.

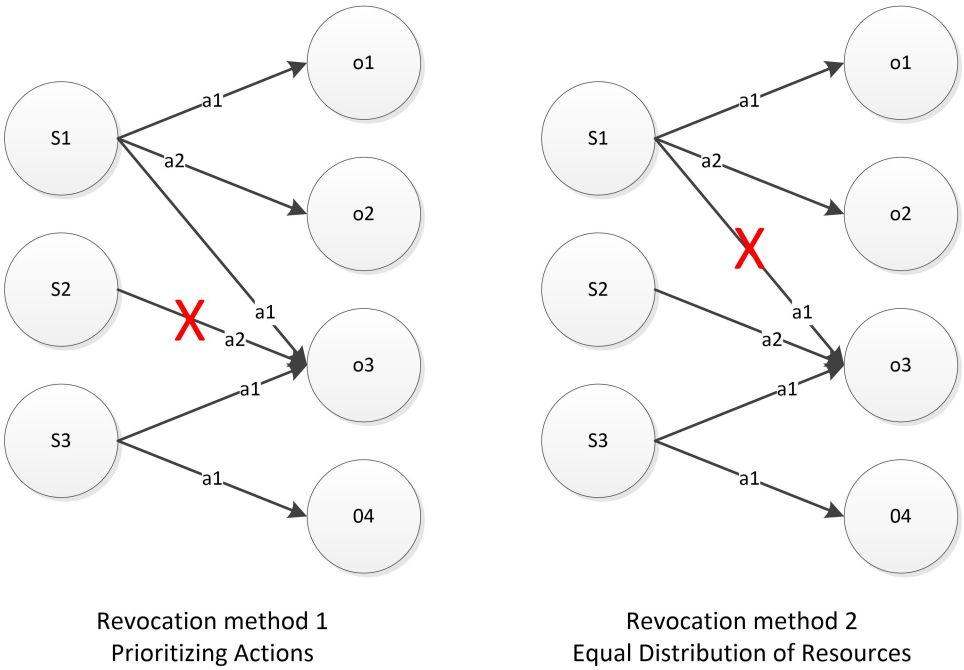

Fig. 4.8 Examples of usage-revoking strategies

In Figure 4.8 we depict two possible running instances of the usage scenario analyzing the operation of usage-revoking. The high-level policy defines an upper limit of two simultaneous usages for every object in a system. In Figure 4.8, a violation of high-level policy exists regarding object 3. By utilizing the first revoking method, a usage requested from subject 2 will be terminated due to the fact that it is requested by an action that has a lower priority (i.e., a2). However, by utilizing the second revocation method, a usage requested from subject 1 will be terminated due to the fact that subject 1 utilizes the largest number in resources in the system.

The revoke-usage strategy is defined with an appropriate *Policy rule* presented in the following next two subsections.

Prioritizing actions

In this usage revoking method actions are organized hierarchically. Therefore, a greater arithmetic value in the attribute of an action, compared with another action, defines a se-

niority relation between them (i.e., the former action is senior to the latter action). Whenever the maximum number of usages on a resource is reached out, the most "junior" usage action in the hierarchy is revoked. The operation of the usage-revoke method entitled *prioritizing actions* is depicted on the left side of Figure 4.8 where usage from *s2* is revoked because it encompasses an action with lower priority *a2*. The semantics of the *Policy rule* that determines which usage must be revoked follows:

$$Policy\_Rule(x,y) \triangleq SelectAction(x).att > SelectAction(y).att$$

where *SelectAction* returns the action that a specific usage exercises. The determination of the specific action is accomplishment through the utilization of the identity attribute value and it is defined as

$$SelectAction(x) \triangleq CHOOSE \quad a \in A : a.id = x.aid$$

It is worthy to mention that the above revoke-usage strategy is *static* since the decision regarding which usage to revoke between two usage is always the same. Thus, the aforementioned decision is independent from the usages that has already been requested or executed in the system.

Equal distribution of resources to subjects

The revoke-usage strategy that supports an equal distribution of resources to the users will, at first, determine the set of subjects that use or request the object that violates the high level policy. In turn, it will revoke the usage that is related with the subject having the greatest number of "activated" resources usages. The operation of the usage-revoke method entitled

equal distribution of resources is depicted on the right side of Figure 4.8 where usage from *s1* is revoked because *s1* occupies the greatest number of objects in the system. The *Policy rule* that implements this revoke-usage strategy follows:

$$Policy\_Rule(x,y) \triangleq NumofUses(SelectSubj(x)) > NumofUses(SelectSubj(y))$$

where *NumofUses(x)* calculates the number of usages that the *direct* subject of usage x currently occupies and it is defined as:

$$NumofUses(x) \triangleq Cardinality(\{u \in U : u.oid = x.oid\})$$

and *SelectSubj(x)* returns the *direct* subject from the usage x and it is defined as:

$$SelectSubj(x) \triangleq CHOOSE \quad s \in S : s.id = x.sid$$

The aforementioned revoke-usage strategy is characterized as *dynamic* since the decision regarding which usage to revoke between two usages, is not always the same. Specifically, the usage revoke decision is based on the number of activated usages of each subject which cannot be predetermined a priori but only during runtime.

4.5 Chapter summary

An access control is proposed in this chapter for ODCE. To meet the requirements posed by ODCE, the proposed UseCON model incorporates a number of significant features compared with existing access/usage control models. Firstly, UseCONs extended expressiveness over the existing usage control models is the result of utilizing information originating

from either a single or a set of both direct and indirect entities in the creation of the usage allowance decision. Secondly, UseCON inherently supports the utilization of historical information of usages through the automatic management of use entities; Moreover, a strict formal model of UseCON is presented in TLA+ that supports the concurrent operation of multiple usages. Lastly, through an example, we demonstrated how to specify policies in the TLA+.

Chapter 5

Evaluation

5.1 Introduction

This chapter elaborates on information regarding the evaluation of the proposed UseCON model. Specifically, Section 5.2 highlights the novel characteristics of the proposed UseCON model which are stemmed from its design decisions. Moreover, in Section 5.3 we present the definition of the properties that should be verified in order to prove the correctness of both the UseCON model and, furthermore, of an example policy that was prior presented in Chapter 4.

5.2 Model characteristics

The UseCON model enhances UCON's fundamental design guidelines as continuity of decision and attribute based usage control. This is done by introducing a number of innovative modeling decisions. Specifically, UseCON directly associates entities with contextual information and also replaces UCON's rights with actions which are enhanced with attributes. Moreover, UseCON utilizes for the usage decision information originating from both *di-*

rect and *indirect* entities. The aforementioned design decisions in combination with the augmented utilization of historical information, which is supported through *uses*, results in enhanced capabilities, as demonstrated in the following examples.

5.2.1 Abstraction of actions

In UCON, rights correspond to permissions for subjects to execute usage functions on objects. However, rights are not described with attributes. The replacement of UCON's simple rights with UseCON's actions described by attributes, provides enhanced capabilities, as follows.

Simplifying the administration of policy rules.

A UCON policy rule governs the allowance either for a specific right or all rights. Thus, every time a new right is introduced in the security system, the policy administrator should most likely create a corresponding policy rule that permits its usage. However, in a computing environment that encompasses a great number of rights, policy administration is becoming a complicated process. In UseCON, the description of actions by attributes provides the policy administrator with the capability to govern the allowance of a set of actions by a single policy rule, as presented in the following example.

Example 1 *A company that offers location discovery services provides the capability to its customers to require the location of an object. A customer, according to his classification, can request the location of an object with a desired accuracy level. For example, members of the "golden" category might request the location of an object with an accuracy expressed in meters, while regular users are able to request the location of an object in kilometers.*

Modeling example 1 in UCON requires the creation of a unique right entity for each accuracy level of the location discovery service. Moreover, the policy administrator must

create an additional policy rule (authorization, condition or obligation) that governs the allowance of the particular right's request. Hence, it is impossible with UCON modeling to create a policy rule that governs the allowance of a subset of rights e.g. rights that model location discovery services.

However, the replacement of UCON rights with UseCON actions associated with attributes provides the capability to model the relation that possibly exists between actions. More specifically, in example 1, every action is associated with an attribute, named *type*. Actions that refer to location discovery services have a unique *type* attribute value e.g. "LocService". Thus, by utilizing the value of *type*, a policy rule is able to govern the allowance of all the actions that represent location discovery services.

The UseCON modeling in example 1, results into the following policy rules rules:

accuracy : $A \rightarrow W$	Location accuracy level supported by the service
category : $S \rightarrow C$	Customer's category. "Premium" or "Regular"
type : $A \rightarrow T$	Type of service. "LocService" for location discovery

$$allowed(s,o,a) \Rightarrow type(a) = \text{“LocService”} \wedge category(s) = \text{“premium”}$$

$$allowed(s,o,a) \Rightarrow type(a) = \text{“LocService”} \wedge category(s) = \text{“regular”} \wedge accuracy(s) = \text{“kilometers”}$$

The first policy rule governs the allowance of two actions (location discovery service with accuracy level of kilometers and location discovery service with accuracy level of meters). The additional accuracy attribute utilized in the second policy rule represents the accuracy level of the location discovery service e.g. "kilometers" or "meters".

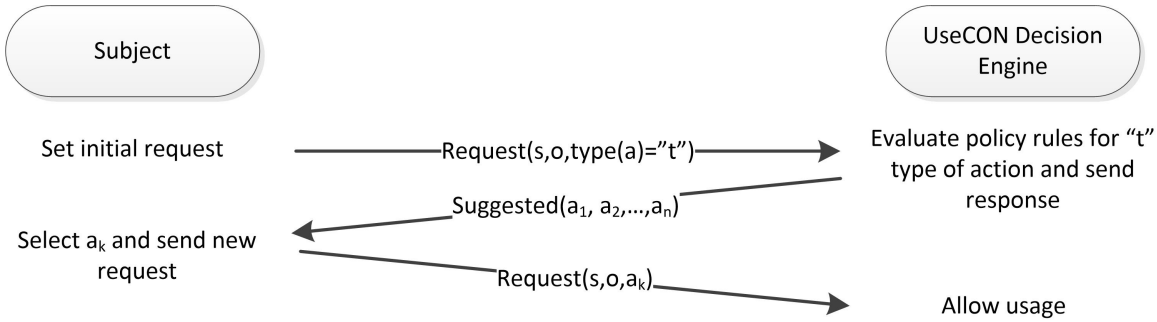

Fig. 5.1 Sequence of messages for action negotiation

Negotiating action parameters.

The utilization of action's attributes in UseCON does not only simplify the policy administration process, as mentioned previously, but also provides enhanced capabilities for negotiating the action parameters of a usage request.

In UCON, a subject is able to request the usage of a specific right but it is not possible to request a "generic" right, e.g. the location of an object without specifying particular accuracy requirements. However, the utilization of the attribute *type*, as introduced in UseCON modeling of example 1, is further able to provide to subjects the capability to request the execution of a usage by only specifying the type of the action. Therefore, the subject of example 1 may request the execution of any action that contains the value “LocService” in the *type* attribute. When the UseCON decision creation engine receives such a request, it evaluates the policy rules that govern the allowance of actions with the specific value in the attribute *type*. Consequently, the usage control system does not respond with a simple allow or deny message, but with a list containing all the suggested actions that the subject is permitted to exercise. Thus, if the returned list is not empty, the subject can select the action that satisfies her needs and send a new request. The sequence of messages exchanged between the subject and the UseCON usage decision engine is depicted in Figure 5.1.

Supporting Action Hierarchies.

Relevant actions can participate in an action hierarchy. An example of action hierarchy in a hospital sector is presented in (Park and Sandhu, 2004) where an action "a doctor writes a remedy on a patient's record" is considered to be senior to the action "a doctor simply reads the patient's medical history". The hierarchy of actions depends on the security policy of the particular usage control system. However, the policy rule for a senior action dominates on the policy rules for all its junior actions. A more detailed example is the following:

Example 2 *The security policy of a hospital defines that only doctors can read the medical history of a patient. However, altering a patient medical record is permitted only to doctors that have the same specialty with the category of the patient's illness.*

As UCON rights are not described with attributes, it seems impossible to model the relations between them and form a hierarchy. In UseCON, however, the classification of actions is possible through the utilization of action attributes. Consequently, both the policy administrator and the usage control mechanism are able to utilize such hierarchy information in order to enhance the expressiveness of the policy rules and to simplify the usage decision creation process, respectively. For example, whenever a subject requests the usage of two directly related actions, a proper usage control mechanism should evaluate only the policy rule that permits the senior action in the action hierarchy. In addition, in UseCON modeling of example 2, the policy administrator is capable of creating a rule that permits the execution of a read action on a medical record of a patient (a junior action), by examining if the requesting subject has previously exercised a write action on the medical record of any patient (a senior action). The modeling of example 2 with the use of a policy rule follows:

snr : $A \rightarrow 2^A$ The set that contains the ids of the senior actions

$$allowed(s,o,a) \Rightarrow | \{u' \in U : status(u') = \text{“completed”} \wedge sid(u') = id(s) \wedge oid(u') = id(o) \wedge aid(u') \in snr(a)\} | \geq 1$$

5.2.2 Utilization of usage information

The introduction of the *use* entity in UseCON provides new capabilities to the policy administrator. The utilization of use entities along with their attributes values provides the capability for enhanced utilization of historical information of usages and proper association of information to the system entities, as it is presented in the following examples.

Supporting Transactions.

Some *properties* are not related with a single entity (subject or object), but with a combination of them. For example, an object attribute in UCON is associating information originating either directly from the object or from the right - object combination e.g. the price of the service (Park and Sandhu, 2004). Thus, if different rights can be exercised on an object, a separate *price* attribute for every one of these rights should be created. In addition, a detailed analysis unveils that the price of a service is actually associating information originating form the subject - object - right combination. More specifically, different customers may be charged with different prices for the execution of the same right on the same object. Therefore, the association of *properties* in a usage control system either with a single entity or with a usage is proposed. The former kind of information is associated with the related entity attributes while the latter with the corresponding use attributes.

While the values of entity attributes are set by an administrative operation, the creation of use entities and their corresponding attribute values are not predetermined but they are accomplished during the operation of the usage control system. More specifically, a subject entity and its attribute values are determined before the execution of any usage. However, a use entity and its attribute values are created only when a subject requests the correspond-

ing usage. The values of use attributes should be assigned with rules that are application dependent and utilize the attribute values of the other entities participating in the usage. An example of information that is associated with use attributes is related to transactions. A transaction is a complicated system process that is composed from a set of particular system usages. In the UseCON model, every usage is modeled through a use entity that is associated with a *transaction* attribute. Uses that belong to the same *transaction* can share the same value of the *transaction* attribute. By utilizing proper values of use attributes, the policy administrator is able to define usage control rules with enhanced expressiveness. An example of the transaction attribute utilization in the creation of the usage decision follows.

Example 3 *In an accounting office the whole set of usages that update the files of a specific customer are forming a transaction. All these usages can be performed by a number of different employees and may concern a number of different files. However, because all these usages belong to the same transaction, they should be covered with the same privacy statement executed once by a single employee.*

In the following policy rule that models example 3 in UseCON, the execution of a *consent* action by any usage of the transaction is examined:

tr : $U \rightarrow T$ The name of the transaction where the usage belongs to

$$allowed(s,o,a) \Rightarrow | \{u' \in U : status(u') = \text{“completed”} \wedge tr(u') = tr(u) \wedge aid(u') = \text{“consent”} \} | \geq 1$$

Enhanced utilization of historical usage information.

Attribute mutability in UCON presents a number of limitations. For example, an attribute update procedure is executed only after the allowance of a requested usage. Thus, the denied

usage requests are not recorded and information regarding such facts is not utilized for subsequent usage decisions. In addition, in an UCON ongoing rule, the same attribute update procedures will be executed if either the usage has been terminated by the subject or revoked by the usage control system, due to the ongoing rule violation. Consequently, UCON is incapable to discriminate the usages terminated by the subject from those revoked by the usage control system.

The UseCON model provides with comprehensive knowledge about the previous system usages through the utilization of the use entity. More specifically, the *state* attribute of a use entity provides the ability to discriminate between requested, active, denied, revoked and terminated usages. Such information can be utilized for future usage decisions. An example, where information about previously revoked usages is used for the creation of the usage control decision follows.

Example 4 *In a Digital Rights Management (DRM) system there is an upper bound limit on the number of simultaneous usages of an object by subjects. Whenever the maximum number of usages of an object is exceeded, several revocation strategies can be applied (Park and Sandhu, 2004). However, as a mean of policy fairness, the execution of a usage that has been previously revoked by the system is freely permitted without the evaluation of additional policy rules.*

The corresponding policy rule that implements the policy described in example 4 follows:

$$allowed(s,o,a) \Rightarrow | \{u' \in U : status(u') = \text{ "revoked" } \wedge sid(u') = sid(u) \wedge aid(u') = aid(u) \wedge oid(u') = oid(u)\} | \geq 1$$

5.3 Model properties

The UseCON model is considered to be a general-purpose, policy-neutral usage control model that is capable of supporting a wide range of policies. However, one of the distinguishing characteristics of UseCON is its capability to inherently support historical information of usages during the usage decision making process through its internal use management process. Therefore, in this section, we define a number of properties that must hold in order to assure the correct operation of the use management process. Moreover, we also define a number of properties for an example of a policy specification that was presented in Section 4.4. In the latter case, we verify the correct operation of the UseCON model regarding a defined high level policy.

5.3.1 Use management

One of the fundamental properties that can be verified in a system is that of type correctness. Specifically, type correctness is considered to be an invariant which determines that all the variables of the system are assigned with values originating only from a specific set of values. The UseCON specification uses a single variable U which corresponds to the set of system uses. The *invariant* property that defines type correctness in the UseCON model, is defined as follows:

$$TypeCorrectness \triangleq U \subseteq Uses$$

where $Uses$ is the set of all records that have the following form (i.e., all the record fields are assigned with values originating from their domains):

$$Uses \triangleq [s.id : SubjectIDS, o.id : ObjectIDS, a.id : ActionIDS, st : USTATE, att : ATTDOMAIN]$$

moreover, the definition of the domains $SubjectIDS, ObjectIDs, ActionIDs$ and $USTATE$ is:

$$SubjectIDs \triangleq \{s.id : s \in S\}$$

$$ObjectIDs \triangleq \{o.id : o \in O\}$$

$$ActionIDs \triangleq \{a.id : a \in A\}$$

$$USTATE \triangleq \{\text{requested, activated, denied, stopped, completed}\}$$

A use is capable of recording detailed historical information about the operation of usages in the system. Consequently, a valid implementation of the UseCON model, where multiple usage processes are operating concurrently, depends on a proper management of the use instances that represent these usages. Specifically, all use instances must adhere only to the state transitions depicted in Figure 4.4 for pre-authorizations or Figure 4.5 for ongoing authorizations. Based on the previous observation, a number of *safety* and *liveness* properties can be defined. For example, a *safety* (nothing bad happens) property states that if a use instance has at any given state the value of "completed" in the *st* attribute, then it must be impossible for the same use instance to have the attribute value of "requested" in the *st* attribute, in a subsequent state. The semantics, expressed in TLA+, which verify the previous property for all the uses of a system are defined by the following temporal formula:

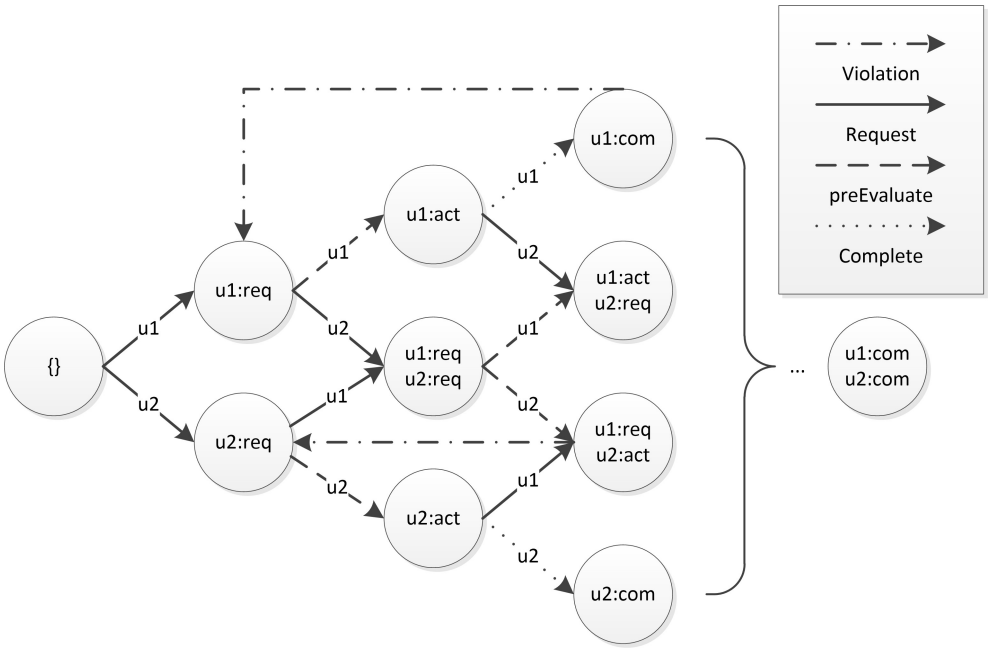

Fig. 5.2 Violation of safety properties

$$Safety \triangleq \forall u \in U : u.st = \text{"completed"} \rightsquigarrow u.st \neq \text{"requested"}$$

Analogous *safety* properties can be defined for all the possible prohibited state transitions. The complete set of prohibited state transitions is depicted in Table 5.1 under the header *"safety"*. Moreover, two examples that illustrate the violation of *safety* properties are depicted in Figure 5.2. Specifically, the transition of the use instance u1 from the state having in the *st* attribute the value of "completed" into the state having in the *st* attribute the value of "requested" results in a violation of the a *safety* property. A similar violation is considered during the transition of the use instance u2 from the state of "activated" into that of "requested".

In addition, the definition of *liveness* properties in the UseCON model determine all the valid state transitions regarding any use instance. For example, Figure 4.4 presents that a use instance that has at any given state an *st* attribute value that is evaluated to "activated" must be followed by an attribute value equal to "completed" in a subsequent state. The TLA+

	Safety		Liveness	
	Former State	**Latter State**	**Former State**	**Latter State**
pre	Completed	any other state	Requested	Activated / Denied
	Activated	Requested or Denied	Requested	Completed / Denied
	Denied	any other state	Activated	Completed
ongoing	Completed	any other state	Requested	Activated
	Activated	Requested	Requested	Completed / Stopped
	Stopped	any other state	Activated	Completed / Stopped

Table 5.1 Safety and liveness properties in UseCON

semantics for defining this property can be defined as follows:

$$Liveness \triangleq \forall u \in U : u.st = \text{"activated"} \rightsquigarrow u.st = \text{"completed"}$$

All the possible state transitions that are eligible to be performed are depicted in Table 5.1 under the header *"Liveness"*.

5.3.2 An example policy

The high level policy described in the usage scenario in subsection 4.4 sets a limit to the permitted maximum concurrent "activated" usages for every object in a system. Consequently, this policy can be verified with an invariant in TLA+, as follows:

$$HighPolicy \triangleq \forall u \in U : NumofActonObject(u) <= MAXLIMIT$$

where *NumofActonObject(u)* returns the number of the usages that are activated on the same object with usage u and it is defined as:

$$NumofActonObject(x) \triangleq Cardinality(\{u \in U : \ \wedge u.oid = x.oid \\ \wedge x.st = "activated"\})$$

The aforementioned high level policy is implemented by two usage revoking methods. The first usage-revoke method utilizes the attribute value of actions in order to categorize usages. Specifically, between two usages the one with the highest action entity value is characterized as "high" level and the other as "low" level. Consequently, in case that there is a policy violation on an object, the usage with the "low" level should be stopped. However, due to the non-determinism imposed by concurrent systems, whenever a "low" level usage (v1) is activated on an object and a "high" level usage (v2) requests to operate on the same object, it is impossible to know if the next action will be a completed action for v1 or an activated action for v2. Therefore, in the first case v1 will be terminated by subject, while in the second case v1 will be stopped by the usage control system. The previous property can be defined in TLA+ using a temporal formula, as follows:

$$\forall v1, v2 \in U : ((\wedge (v1.oid = v2.oid \wedge Policy\_Rule(v1, v2)) \\ \wedge (v1.st = "activated" \wedge v2.st = "requested")) \rightsquigarrow \\ (\vee (v1.st = "completed" \wedge v2.st = "requested") \\ \vee (v1.st = "stopped" \wedge v2.st = "activated")))$$

where *Policy_Rule* categorizes usages v1, v2 to "high" and "low" level, respectively. The same property can be utilized for the verification of the second usage-revoke method. The only difference between the two property definition is that the second usage-revoke method

is the *Policy_Rule* which now should evaluate the number of activated usages as it is described in subsection 4.4.

5.4 Chapter summary

In this chapter, we presented the unique characteristics of the UseCON model. Moreover, having a formal specification of the UseCON model in TLA+, we defined a set of properties that fall under the categories of type consistency, safety and liveness. These properties have been defined for both the correct operation of use management performed by the internal operations of UseCON, and therefore, can be checked for their correctness i.e, the adherence to the initially defined requirements of the UseCON model. In addition, we defined a set of properties in order to verify the correct enforcement of the policy example presented in Chapter 4 according to the UseCON model.

Chapter 6

Formal verification

6.1 Introduction

The application of model checking as a formal verification method has as an advantage the existence of a variety of model checking software tools that eliminates the requirement for the development of a specialized tool for the verification of the defined properties. Therefore, in this chapter, we provide information regarding the verification of the properties defined in Chapter 5 using the TLA+ model checker (TLC) software, which is included in the TLA+ Toolbox (*toolbox*). *Toolbox* is an Integrated Development Environment (IDE), designed for the definition and verification of TLA+ specifications, which includes a set of integrated TLA+ Tools (TLA, 2013). Thereafter, this chapter continues in Section 6.2 with the presentation of *toolbox* fundamentals along with the implementation details for the verification of both the use management and of the policy example properties that were defined in Chapter 5. Moreover, Section 6.3 elaborates on information regarding the verification results provided by the TLC model checker.

6.2 Model checking with TLC

Toolbox is an Integrated Development Environment (IDE), which is designed for the definition and verification of TLA+ specifications (TLA, 2013). Specifically, the *toolbox* editor provides functionality for the definition and alteration of TLA+ specifications, and supports syntax highlighting. Additionally, an automatic parser checks the defined specifications for syntax errors and presents them accordingly by marking them in the used modules. The *toolbox* IDE also supports printing of specifications in a pretty form via the pretty-printer tool.

The tool responsible for the verification of a TLA+ specification in *toolbox* is the TLC model checker. In particular, TLC explicitly generates and computes all the possible states of a system. However, many times the specification of a system might contain an infinite number of states. TLC handles such specifications, by choosing a finite model of the system and in turn checks it thoroughly. Specifically, the creation of a system's model in TLC requires the definition of its specifications, properties and values of constant parameters (TLA, 2014). A specification represents all the behaviors that have to be checked. Moreover, the values assigned to constant parameters are utilized for the instantiation of a specification. TLC can check a model for deadlocks, invariants and properties (TLA, 2014). A deadlock is occurred when the model reaches to a state in which its next-state action allows no successor states. However, a deadlock is not considered always to be an undesirable property since in some systems is considered to be a desired termination property. An invariant is a state predicate as it is described in Section 2.5. Invariants are evaluated to true if they are valid for all the reachable states of the system. Properties are temporal formulas that must be evaluated to true for all the behaviors of the model. TLC has some limitations regarding the handling of a subclass of TLA+ specifications and properties that it can check like the incapability to support the action composition operator (Lamport, 2002). A very helpful

feature of TLC is the fact that for all cases it identifies an error during the verification process, TLC provides an error trace viewer that allows the exploration in a structured view of the debugging information. Moreover, TLC supports an arbitrary evaluation of states and action formulas in each step of a trace.

6.2.1 Use management

The TLC model checker requires for the process of model checking defining the specification of the model and the properties to be verified. However, in order to define a model in TLC, e.g. the model of UseCON, it is required to declare the model's specification along with the values of constant parameters. The constant parameters of the UseCON model consist of records assigned with values using a TLA+ specification. An example of the TLA+ specification that declares subject entities of the UseCON model follows:

$$s1 \triangleq [id \mapsto 1, att \mapsto 10]$$

$$s2 \triangleq [id \mapsto 2, att \mapsto 20]$$

$$s3 \triangleq [id \mapsto 3, att \mapsto 30]$$

where, three subjects ($s1, s2, s3$) are declared to have each of them an identification attribute (id), assigned with a unique value, and a single generic attribute (att) assigned with a constant value. In a similar way, we can declare both action and object entities.

Another requirement for the complete specification of the UseCON model, is the definition of the policy rules that govern the allowance of a usage request. The specification contains three types of policy rules. The first policy rule allows the operation of any requested usage. The second policy rule decides randomly the operation of a usage request

based on the TLA+ function *RandomElement*. The third policy rule decides the operation of a requested usage, only if the value of the subject attribute is less than the value of the object attribute. The semantics of the aforementioned policy rules, follows:

$$1) Policy\_Rule1(x,y) \triangleq TRUE$$

$$2) Policy\_Rule2(x,y) \triangleq RandomElement(\{TRUE, FALSE\})$$

$$3) Policy\_Rule3(x,y) \triangleq SelectSubject(x).att < SelectObject(x).att$$

$$SelectSubject \triangleq CHOOSE\, s \in S : s.id = x$$

$$SelectObject \triangleq CHOOSE\, o \in O : o.id = x$$

The last requirement for the operation of the model checking process with TLC is that of defining the properties to be examined. The first property is that of the *TypeCorrectness* invariant as it is described in subsection 5.3.1. Furthermore, all the safety and liveness properties are declared, as these are presented in Table 5.1. Due to limitations of the TLC model checker, the safety and liveness properties are declared for every usage separately, (e.g., the usage 1 is related to object *o1*, subject *s1* and action *a1*). The semantics of the safety and liveness properties, of e.g. usage 1, follows:

$$Liveness1 \triangleq (useRequested \in U) \rightsquigarrow ((useCompleted \in U) \vee (useDenied \in U))$$

$$Safety1 \triangleq (useCompleted \in U) \rightsquigarrow (useRequested \notin U)$$

where *useRequested* denotes the usage request of the tuple $< s1, a1, o1 >$ with semantics as follows:

$$useRequested \triangleq createUse(<< s1, a1, o1 >>)$$

In an analogous manner the semantics of *useDenied*, *useActivated*, *useStopped*, *useCompleted*, follows:

$$useActivated \triangleq preUpdate(useRequested)$$

$$useDenied \triangleq denUpdate(useRequested)$$

$$useCompleted \triangleq comUpdate(useRequested)$$

$$useStopped \triangleq stopUpdate(useRequested)$$

The declaration of the pre UseCON model in TLC is depicted in figure 6.1. The name of the specification is *Spec* and there is no additional declaration of the constant parameters since they have already been declared in the specification (see Appendix). The *TypeCorrectness* invariant is checked along with the *Liveness* and *Safety* properties for every usage. Moreover, the specification is not checked for deadlock (the explanation for the deadlock verification avoidance is explained in subsection 6.3). The declaration of the ongoing UseCON model in TLC is analogous since it utilizes the same constant parameters and requires the declaration of analogous properties. The verification results of both pre and ongoing UseCON model are presented in subsection 6.3

6.2.2 An example policy

Analogous definitions for the specification, constant parameters and verified properties are required to perform the model checking process for the policy example presented in subsection 4.4. However, the constant parameters are the same as in the use management case,

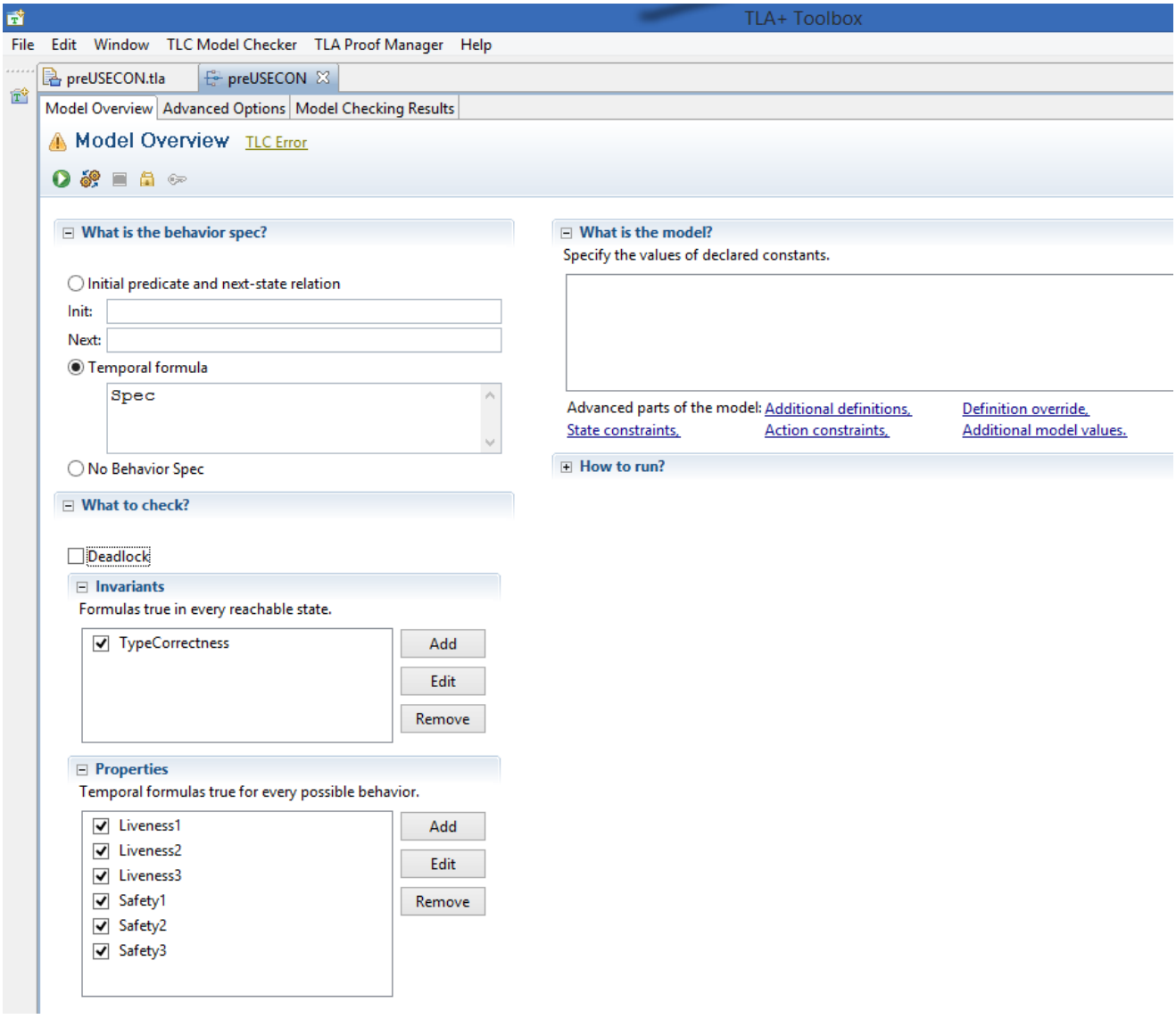

Fig. 6.1 Model declaration in TLC

and therefore, they are not presented. Moreover, the *Type Correctness* invariant is also the same as the use management since it is a generic property in any UseCON implementation. Additionally, a new property that is valid for the policy example is that of *High Policy* property, which verifies that every resource of the system has a maximum number of concurrent "activated" usages with the following semantics:

$$HighPolicy \triangleq \forall\, u \in U : NumofActonObj \leq 1$$

where the semantics of *NumofActonObj* are described in subsection 5.3. We adopt one as the maximum number of usages due to the restriction imposed by the state explosion problem as it is described in the following subsection 6.3.

Another property declared in the application example is the correct operation of the usage revoking method as it is described in subsection 5.3. The semantics of this property for the first usage revoke method, follows:

$$\begin{aligned} RevokePolicy \triangleq & (useActivated(u1) \in U \wedge useRequested(u2) \in U) \rightsquigarrow \\ & ((useStopped(u1) \in U \wedge useActivated(u2) \in U) \vee \\ & (useCompleted(u1) \in U \wedge useActivated(u2) \in U)) \end{aligned}$$

since it is known a priori that usage $u2$ has a higher priority than that of usage $u1$. However, the definition of the second usage revoke method requires the evaluation of the *policy rule* to verify the priority among usages. Consequently, the semantics of the property related to the second revoking method, is defined as follows:

$$RevokePolicy \triangleq$$

$$(useActivated(u1) \in U \wedge useRequested(u2) \in U \wedge Policy\_Rule(u1,u2)) \rightsquigarrow$$

$$((useStopped(u1) \in U \wedge useActivated(u2) \in U) \vee$$

$$(useCompleted(u1) \in U \wedge useActivated(u2) \in U))$$

It is worth mentioning that the verification of the second revoke policy property can be applied only on usages $u1$ and $u2$. The aforementioned restriction is implied due to the non-determenistic nature of the model. Specifically, it is not ensured that, after the evaluation of the first *policy Rule*, a new request of a third $u3$ usage would not be able to alter the outcome of the *Policy Rule* in a future state.

6.3 Discussion

The model checking verification was conducted in a platform with the following technical characteristics: 3rd generation 2GHz Intel i7 CPU, 8 GByte of RAM, and using the Windows 8.1 Professional (64bit version) operating system. TLC's version was 2.05.

The first verification results conducted from testing the deadlock property in the UseCON system. Specifically, for the pre authorization UseCON model, we considered all the usages of the model to be requested, and therefore, be in the state of "activated" or "denied". The usages being activated were finally completed. Therefore, the final state in every usage must be "denied" or "completed". The TLC model checker evaluates all the behaviors of the model and terminates when it reaches to a deadlock. Moreover, the actions of the deadlocked behavior are presented together with the attribute values in each state. Figure 6.2 depicts the results of a deadlock behavior with the values of variable U in the bottom. No-

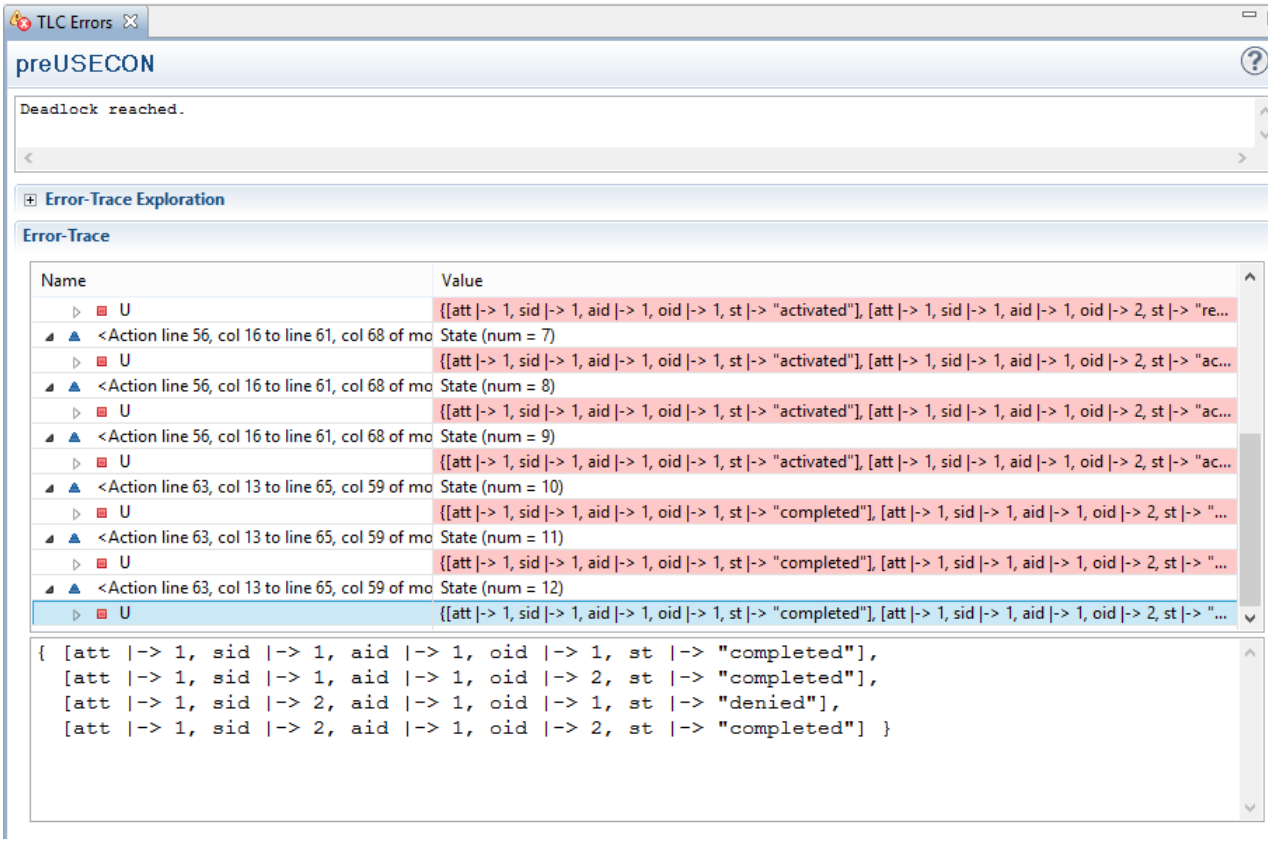

Fig. 6.2 Deadlock violation results in TLC

tice that every usage state is "completed" or "denied".

Moreover, we performed a verification test on the same model but without checking the *deadlock* property. The verification was finalized without presenting any errors. The same process repeated for all the defined policy rules. The results are depicted in figure 6.3.

The analysis of the verification results require to know the internal procedures that TLC is making use in order to compute all the possible behaviors of the model (i.e., creation of the transition system). Initially, TLC computes the states that verifies the *Init* predicate and inserts them into a set G. For every state $s \in G$ TLC computes all the possible states t that $s \mapsto t$ can be a step in a behavior. Specifically, TLC substitutes the values assigned to variables by state s for the unprimed variables of the *Next* action, and then it computes all the possible assignment of values to the primed variables that makes the *Next* action

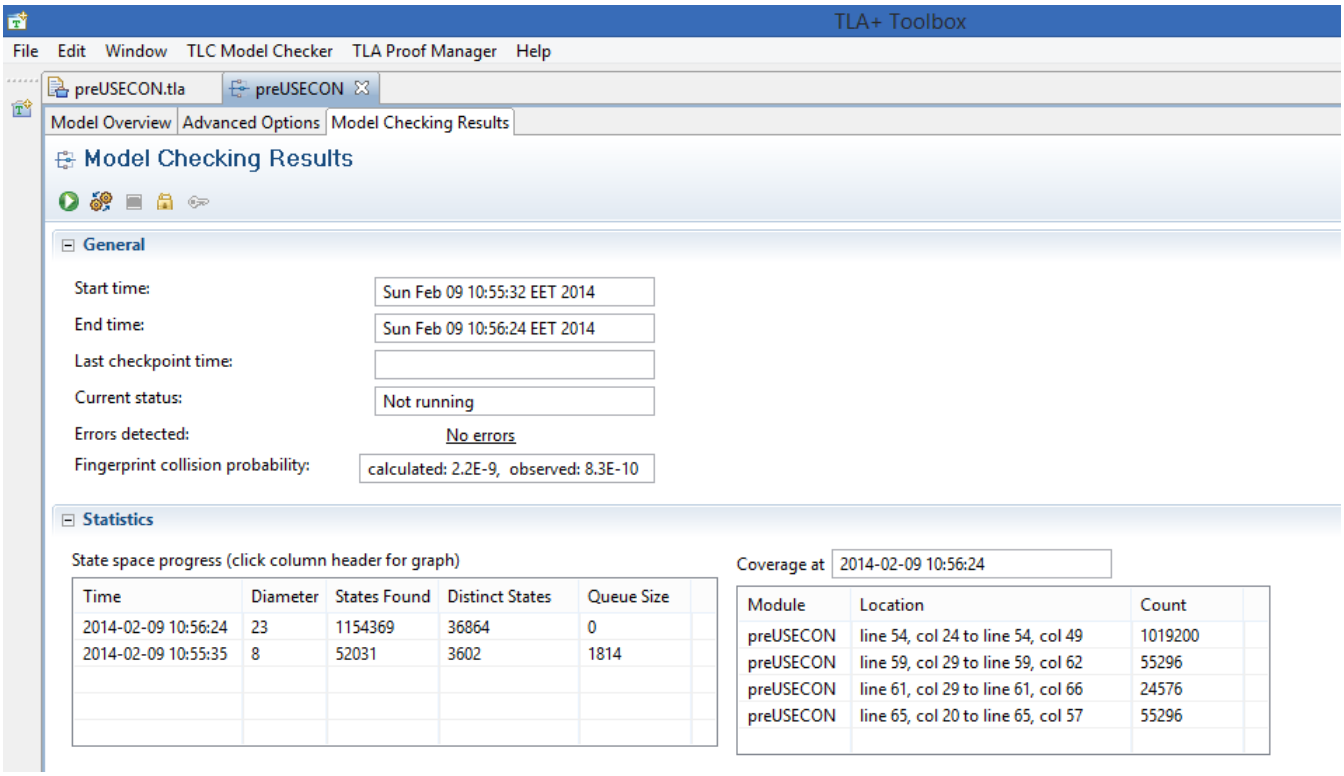

Fig. 6.3 Verification results in TLC

true. For every state t found by the former procedure it is added to set G if it is does not already exists. The previous two actions are repeated until no new states can be added in G. Therefore, the verification results produced by TLC, are the following:

- Diameter: The number of states in the longest path of G in which no state appears twice
- States Found: The number of states examined.
- Distinct States: The number of distinct states examined.

The verification results produced by TLC for the pre and ongoing UseCON models are presented in table 6.1. An additional column presents the actual running time of the TLC model checker presented in seconds.

	Usages	Diameter	States Found	Distinict States	Time
pre UseCON	1	3	4	4	1
	2	7	19	15	1
	4	13	1685	571	2
	6	19	96181	15382	8
	8	25	4883306	389860	306
ongoing UseCON	1	4	5	4	1
	2	7	36	20	1
	4	13	2355	613	1
	6	19	116743	15592	11
	8	25	5518759	390538	503

Table 6.1 Verification results for UseCON models

The application of the model checking technique in this dissertation provided a formal verification of the properties for both the use management of the UseCON model along with the policy example. Even, the trace of the deadlock "error" verifies that the defined specification of the system operates in a correct manner. One of the advantages of using model checking techniques for the verification of models is that it is able to provide proofs of correctness without requiring an implementation of the model. On the other side, a disadvantage of applying model checking is that it is prone to the state explosion problem. Specifically, in our case, the verification of the defined properties is time consuming when it is required to verify more than 8 usages in the UseCON model. This was also depicted in Table 6.1. However since we have checked the correctness of the model for less than 8 usages in combination with the fact that an addition of a usage does not affect the specification of the model leads to the conclusion that the properties are also valid in models using any number of usages. Moreover, problems were also originated by limitations of the model checking software (e.g., it is impossible to define a property that includes both the forall $\forall$ and a temporal operator). Another issue that we identified is the verification of ongoing policy rules. Specifically, a usage request might lead to a violation of the policy

rule in another usage. Consequently, any usage request should be followed by an evaluation of the policy rules for all the usages in the system. Finally, we conclude that the area of verifying ongoing models is challenging because of non-determinism regarding ongoing system actions and might be an area for further examination.

6.4 Chapter summary

In this chapter, we shortly presented the TLA+ Toolbox that we used as an IDE for the verification of the properties specified in Chapter 5 using the TLC model checker. In turn, we provided information on the declaration of models in TLC as well as a series of performance metrics. Furthermore, we elaborated in the internal procedures of the TCL model checker. Therefore, we have proved the adherence of the defined model to the initially defined requirements using an error-free model checking technique.

Chapter 7

Conclusions

In this dissertation, we have investigated access control in the context of ODCE. In particular, introductory information and related work are presented in Chapters 1-2. Chapter 3, provided an analysis of ODCE that eventually led as to the identification of the access control requirements in that type of computing environments. In Chapter 4, we proposed our new attribute based usage control approach for ODCE along with the definition of a policy example specification. An evaluation of the proposed model in terms of the presented characteristics and properties are presented in Chapter 5. In Chapter 6, we elaborated on the verification of the proposed model and the presented policy example using a model checking technique, and a discussion of the results. In this last chapter, we shortly summarize the contributions of this dissertation and we outline some topics left for future work.

7.1 Summary of the contributions

The summary of the contributions of this dissertation are the following:

A detailed study of the access control requirements of access control in ODCE is presented through motivating scenarios and related literature. Specifically, support of partially

unknown users, cooperation among heterogeneous entities, utilization of contextual information, protection of privacy, ease of administration, resource constrained operations and adaptation on operational changes are recognized as essential access control requirements in ODCE. Moreover, a number of challenging issues, which are faced when UCON is applied in ODCE, are highlighted through the utilization of representative usage scenarios. The results of this study are revealing various limitations in contextual information handling, lack to support complicated usage modes of subjects on objects, and weaknesses in utilizing information concerning previous or current usages of system resources. Moreover,

Based on the requirements recognized in the previous step, a new Use-based usage CONtrol (UseCON) approach that supports recording of usages with the help of a new entity, entitled use, is presented. The proposed approach provides enhanced contextual information modeling, support of complicated access modes and an alternative approach in obligations modeling.

The UseCON model is characterized by extended expressiveness when compared with existing approaches. Specifically, when it comes in decision making, the UseCON model can use information from both direct and indirect entities. Furthermore, decision making in UseCON can be based also on information regarding a subset of entities (i.e., cardinality or sum of properties). Another design decision that enhances the expressiveness of UseCON stems from the fact that it is able to use detailed historical information from usages (e.g., usages that have been rejected) in contrast with existing solutions (e.g., UCON). This results in the description of policies that cannot be supported by current access control approaches.

Moreover, in UseCON the substitution of UCON rights with actions described with attributes leads to the support of new capabilities as the ease of policy management and the negotiation of action parameters (see subsection 5.2.1).

The utilization of historical usage information is inherently supported by UseCON and not through an attribute mutability mechanism as in the UCON model. Consequently, the

policy administrator can use historical information for the decision making process without requiring any further actions. This leads to the fact that the definition of policies in UseCON is simpler that in other approaches and even closer to the natural language.

Finally, we presented a formal specification of the UseCON model in TLA+ and provided a sound and solid formal definition of the internal actions performed by the model. The proposed formal model is characterized by remarkable clarity stemmed from its initial design decisions. In addition, formal model of UseCON it is capable to support the concurrent operation of multiple usages along with the utilization of information originating from *indirect* entities.

The presented UseCON model was evaluated not only in terms of the presented characteristics but also with the definition of desired properties. Specifically, type consistency, safety and liveness properties that verify the correct utilization of historical information by the core model and its internal operations were verified for their correctness (i.e., adherence of the defined model to the initially defined requirements) through the application of an automated and error-free model checking technique.

7.2 Future work

The research described in this dissertation can be extended along several directions.

Prototype implementation. Our proposed attribute based usage control model has been formally defined and verified for a set of properties. The same applies for some of the models that were capable of enforcing usage control policies, and presented in Chapter 2. However, to the best of our knowledge, none of the usage control models has been implemented. Usage control introduces a higher level of complexity compared with RBAC and ABAC when it comes to its implementation because of the existence of continuity of decision and of pre and post rules. Therefore, a challenging problem for future work might

be the analysis of the complexities posed by usage control models and an implementation of a mechanism able to enforce such policies.

Usage control in multi-domain systems. In most systems, access control decisions are taken on a per domain basis. However, recent advances in distributed and collaborative systems (e.g., Grid and cloud computing systems) require the collaboration among several domains, where each one implements its own security policy. In the case of multi-domain access control several problems arise as secure inter-operation (Gouglidis et al., 2013). Therefore, it would be interesting also to examine the application and the security issues that might exist in the case of applying a usage control model in such collaborative environments.

Property verification. The proposed access control model has been formally verified using an automated and error-free model checking technique. During the process of verifying properties, we have noted that when we apply a large number of usages we come up against a combinatorial blow up of the state space, which is commonly known as the state explosion problem. Therefore, it might be also interesting to check the correctness of a model using other approaches as a formal proof management system. An example of such a system is Coq, which provides a formal language to write mathematical definitions, executable algorithms and theorems together with an environment for semi-interactive development of machine-checked proofs (Coq, 2014).

7.3 Closing remarks

This work has appeared in varying forms of conferences (see appendix B). Particularly, a set of new access control requirements were examined in (Grompanopoulos and Mavridis, 2010). Our proposed usage control model and critiques on existing approaches were presented in (Grompanopoulos and Mavridis, 2012a,b; Grompanopoulos et al., 2013). A for-

mal representation of the UseCON along with the presentation of a verification technique for both the internal operations of the proposed model and its policies were proposed in (Grompanopoulos et al., 2013).

Appendix A

TLA+ source code

UseCON pre-authorization model

```
---------------------------- MODULE preUSECON ----------------------------
EXTENDS Integers, FiniteSets, TLC, Reals

VARIABLES U

\*      DECLARATION OF CONSTANT VALUES FOR S,A,O
\* (*************************************************************************
s1== [ id |-> 1, att |-> 10]
s2== [ id |-> 2, att |-> 20]
s3== [ id |-> 3, att |-> 30]
s4== [ id |-> 4, att |-> 40]
s5== [ id |-> 5, att |-> 50]

a1== [ id |-> 1, att |-> 11]
```

```
a2== [ id |-> 2, att |-> 21]

a3== [ id |-> 3, att |-> 31]


o1== [ id |-> 1, att |-> 12]

o2== [ id |-> 2, att |-> 22]

o3== [ id |-> 3, att |-> 32]


\*      THE SETS OF ENTITIES

S == {s1,s2}

A == {a1}

O == {o1,o2}


----------------------------------------------------------------------------


\*      POLICY RULES (RANDOM DOES NOT PRESENT TRACE TLC BUG)

\* (************************************************************************

SelectSubject(x) == CHOOSE s \in S : s.id=x

SelectObject(x) == CHOOSE o \in O: o.id=x


Policy_Rule(x,y) == SelectSubject(x).att < SelectObject(y).att

\* Policy_Rule(x,y) == RandomElement({TRUE,FALSE})


\*      USE UPDATE ATTRIBUTES

\* (************************************************************************

createUse(x) == [sid|->x[1].id, aid|->x[2].id, oid|->x[3].id, st|->"requested",

att|->1]
```

```
preUpdate(x) == [x EXCEPT !.st="activated"]

onUpdate(x) == [x EXCEPT !.st="activated"]

denUpdate(x) == [x EXCEPT !.st="denied"]

termUpdate(x) == [x EXCEPT !.st="terminated"]

comUpdate(x) == [x EXCEPT !.st="completed"]



\*        PRE AUTHORIZATION MODEL (WITH USES # CONSTRAINT)

\* (***************************************************************************

Init == U = {}

vars == <<U>>


Request ==  \E u \in (S \X A \X O):

                    /\ \forall x \in U : (x.sid#u[1].id \/

x.aid#u[2].id \/ x.oid#u[3].id)

                    /\ U' = U \cup {createUse(u)}

                    \* UNION { U, {createUse(u)}} )


preEvaluate == \E u \in U:  (

                    /\ u.st="requested"

                    /\ IF (Policy_Rule(u.sid,u.oid)) THEN

                            U' = (U \ {u}) \cup {preUpdate(u)}

                    \* UNION { (U\{u}) , {preUpdate(u)}}

                       ELSE

                            U' = UNION { (U\{u}) , {denUpdate(u)}} )
```

```
Complete == \E u \in U:  (
                /\ u.st="activated"
                /\ U' = UNION { (U\{u}) , {comUpdate(u)}} )

Next == Request \/ preEvaluate \/ Complete

Spec == Init /\ [][Next]_vars /\ WF_vars(Next)

----------------------------------------------------------------------------

\*      TYPE CORRECTNESS
\* (************************************************************************
USTATE == {"requested","activated","denied","terminated","completed"}

SubjectIDs == { s.id : s \in S }
ActionIDs == { a.id : a \in A }
ObjectIDs == { o.id : o \in O }
Entities == [ sid : SubjectIDs,
              aid : ActionIDs,
              oid : ObjectIDs,
              st : USTATE,
              att : {1} ]
TypeCorrectness == U \subseteq Entities

\*      SAFETY - LIVENESS
\* (************************************************************************
```

```
entityRequested == createUse(<<s1,a1,o1>>)

entityActivated == preUpdate(entityRequested)

entityCompleted == comUpdate(entityRequested)

entityDenied == denUpdate(entityRequested)



Liveness1 == (entityRequested \in U) ~> ( (entityActivated \in U) \/

(entityDenied \in U) )

Liveness2 == (entityRequested \in U) ~> ( (entityCompleted \in U) \/

(entityDenied \in U) )

Liveness3 == (entityActivated \in U) ~> ( entityCompleted \in U )


Safety1 == (entityCompleted \in U) ~> ( ( entityRequested \notin U )

/\ ( entityActivated \notin U) /\ (entityDenied \notin U) )

Safety2 == (entityActivated \in U) ~> ( ( entityRequested \notin U ) /\

(entityDenied \notin U) )

Safety3 == (entityDenied \in U) ~> ( ( entityRequested \notin U ) /\

( entityActivated \notin U) /\ (entityCompleted \notin U) )



================================================================================

\* Modification History

\* Last modified 2014 by groban

\* Created  2014 by groban
```

UseCON ongoing authorization model

```
---------------------------- MODULE onUSECON1 ----------------------------

EXTENDS Integers, FiniteSets, TLC, Reals


VARIABLES U

\* DECLARATION OF CONSTANT VALUES FOR S,A,O

\* (***************************************************************************

s1== [ id |-> 1, att |-> 10]

s2== [ id |-> 2, att |-> 20]

s3== [ id |-> 3, att |-> 30]

s4== [ id |-> 4, att |-> 40]

s5== [ id |-> 5, att |-> 50]


a1== [ id |-> 1, att |-> 11]

a2== [ id |-> 2, att |-> 21]

a3== [ id |-> 3, att |-> 31]


o1== [ id |-> 1, att |-> 12]

o2== [ id |-> 2, att |-> 22]

o3== [ id |-> 3, att |-> 32]


\* THE SETS OF ENTITIES

S == {s1,s2,s3,s4}

A == {a1,a2}

O == {o1}
```

```
----------------------------------------------------------------------------

\* POLICY RULES (RANDOM DOES NOT PRESENT TRACE TLC BUG)
\* (************************************************************************
SelectSubject(x) == CHOOSE s \in S : s.id=x
SelectObject(x) == CHOOSE o \in O: o.id=x

\* Policy_Rule(x,y) == SelectSubject(x).att < SelectObject(y).att
Policy_Rule(x,y) == RandomElement({TRUE,FALSE})

\* USE UPDATE ATTRIBUTES
\* (************************************************************************
createUse(x) == [sid|->x[1].id, aid|->x[2].id, oid|->x[3].id, st|->"requested",
att|->1]
preUpdate(x) == [x EXCEPT !.st="activated"]
onUpdate(x) == [x EXCEPT !.st="activated"]
denUpdate(x) == [x EXCEPT !.st="denied"]
termUpdate(x) == [x EXCEPT !.st="terminated"]
comUpdate(x) == [x EXCEPT !.st="completed"]


\*        ONGOING AUTHORIZATION MODEL (WITH USES # CONSTRAINT)
\* (************************************************************************
Init == U = {}
vars == <<U>>
```

```
Request ==  \E u \in (S \X A \X O):
                    /\ \forall x \in U : (x.sid#u[1].id \/
x.aid#u[2].id \/ x.oid#u[3].id)
                    /\ U' = U \cup {createUse(u)}

Activate == \E u \in U:  (
                /\ u.st="requested"
                /\ U' = (U \ {u}) \cup {preUpdate(u)} )

onEvaluate == \E u \in U:  (
                    /\ u.st="activated"
                    /\ IF (Policy_Rule(u.sid,u.oid)) THEN
                            U' = (U \ {u}) \cup {preUpdate(u)}
                       ELSE
                            U' = (U \ {u}) \cup {termUpdate(u)} )

Complete == \E u \in U:  (
                /\ u.st="activated"
                /\ U' = (U \ {u}) \cup {comUpdate(u)} )

Next == Request \/ Activate \/ onEvaluate \/ Complete

Spec == Init /\ [][Next]_vars /\ WF_vars(Next)

----------------------------------------------------------------------------
```

```
\* TYPE CORRECTNESS
\* (***************************************************************************
USTATE == {"requested","activated","denied","terminated","completed"}
SubjectIDs == { s.id : s \in S }
ActionIDs == { a.id : a \in A }
ObjectIDs == { o.id : o \in O }
Entities == [ sid : SubjectIDs,
              aid : ActionIDs,
              oid : ObjectIDs,
              st : USTATE,
              att : {1} ]
TypeCorrectness == U \subseteq Entities

\* SAFETY - LIVENESS
\* (***************************************************************************
entityRequested == createUse(<<s1, a1, o1>>)
entityCompleted == comUpdate(entityRequested)
entityActivated == preUpdate(entityRequested)
entityTerminated == termUpdate(entityRequested)

Liveness1 == (entityRequested \in U) ~> (entityActivated \in U)
Liveness2 == (entityRequested \in U) ~> ( (entityCompleted \in U) \/
(entityTerminated \in U) )
Liveness3 == (entityActivated \in U) ~> ( (entityCompleted \in U) \/
(entityTerminated \in U) )
```

```
Safety1 == (entityCompleted \in U) ~> ( ( entityRequested \notin U )
/\ ( entityActivated \notin U) /\ (entityTerminated \notin U) )
Safety2 == (entityActivated \in U) ~>  ( entityRequested \notin U )
Safety3 == (entityTerminated \in U) ~> ( ( entityRequested \notin U )
/\ ( entityActivated \notin U) /\ (entityCompleted \notin U) )
=============================================================================
\* Modification History
\* Last modified  2014 by groban
\* Created 2014 by groban
```

Policy example with revoke usage method 1

```
------------------------------ MODULE Policy1 ------------------------------

EXTENDS Integers, FiniteSets, TLC, Reals


VARIABLES U

\* DECLARATION OF CONSTANT VALUES FOR S,A,O

\* (***************************************************************************

s1== [ id |-> 1, att |-> 10]

s2== [ id |-> 2, att |-> 20]

s3== [ id |-> 3, att |-> 30]

s4== [ id |-> 4, att |-> 40]

s5== [ id |-> 5, att |-> 50]


a1== [ id |-> 1, att |-> 11]

a2== [ id |-> 2, att |-> 21]

a3== [ id |-> 3, att |-> 31]

a4== [ id |-> 4, att |-> 41]

a5== [ id |-> 5, att |-> 51]


o1== [ id |-> 1, att |-> 12]

o2== [ id |-> 2, att |-> 22]

o3== [ id |-> 3, att |-> 32]

o4== [ id |-> 4, att |-> 42]

o5== [ id |-> 5, att |-> 52]
```

```
\* THE SETS OF ENTITIES
S == {s1,s2}
A == {a1,a2}
O == {o1,o2}

----------------------------------------------------------------------------

\*  POLICY RULES (JUNIOR ACTION IN SAME OBJECT)
\* (***************************************************************************
SelectAction(x) == CHOOSE a \in A: a.id=x.aid
Policy_Rule(x,y) == SelectAction(x).att > SelectAction(y).att

\* Finds the # of activated uses with same object as u
SameObjectAct(x,y) == x.oid = y.oid /\ x.st="activated"
ActivatedOnObject(x) == {u \in U: SameObjectAct(u,x)}
NumOfActOnObject(x) == Cardinality(ActivatedOnObject(x))

\* USE UPDATE ATTRIBUTES
\* (***************************************************************************
createUse(x) == [sid|->x[1].id, aid|->x[2].id, oid|->x[3].id, st|->"requested",
att|->1]
preUpdate(x) == [x EXCEPT !.st="activated"]
onUpdate(x) == [x EXCEPT !.st="activated"]
denUpdate(x) == [x EXCEPT !.st="denied"]
termUpdate(x) == [x EXCEPT !.st="terminated"]
comUpdate(x) == [x EXCEPT !.st="completed"]
```

```
\*  POLICY IMPLEMENTATION
\* (***************************************************************************
Init == U = {}
vars == <<U>>

Request ==  \E u \in (S \X A \X O):
                /\ \forall x \in U : (x.sid#u[1].id \/ x.aid#u[2].id \/
x.oid#u[3].id)
                /\ U' = U \cup {createUse(u)}

preEvaluate == \E u \in U:  (
                    /\ u.st="requested"
                    /\  \/  (   /\ NumOfActOnObject(u) < 1
                    /\ U' = (U \ {u}) \cup {preUpdate(u)}
                    )
                    \/  (   /\ NumOfActOnObject(u) >= 1
                           /\ LET act==CHOOSE x \in U : SameObjectAct(x,u)
                  IN
               IF (Policy_Rule(act,u))
                THEN
                    U' = (U \ {u}) \cup {termUpdate(u)}
                ELSE
                    U' = (U \ {u,act}) \cup {preUpdate(u),termUpdate(act)}
                           )
```

```
                    )

Complete == \E u \in U:  (
                /\ u.st="activated"
                /\ U' = (U \ {u}) \cup {comUpdate(u)} )

Next == Request \/ preEvaluate \/ Complete

Spec == /\ Init /\ [][Next]_vars /\ WF_vars(Next)

-----------------------------------------------------------------------------

\* TYPE CORRECTNESS
\* (*************************************************************************
USTATE == {"requested","activated","denied","terminated","completed"}

SubjectIDs == { s.id : s \in S }
ActionIDs == { a.id : a \in A }
ObjectIDs == { o.id : o \in O }
Entities == [ sid : SubjectIDs,
              aid : ActionIDs,
              oid : ObjectIDs,
              st : USTATE,
              att : {1} ]
TypeCorrectness == U \subseteq Entities
```

```
\* HIGH LEVEL PROPERTIES
\* (*************************************************************************
u1 == createUse(<<s1, a1, o1>>)
u2 == createUse(<<s1, a2, o1>>)

test == Policy_Rule(u2,u1)
NewInt2 == \forall u \in U : NumOfActOnObject(u)<=1
NewInt3 ==  (preUpdate(u1) \in U /\ u2 \in U) ~>
( (termUpdate(u1) \in U) /\ (preUpdate(u2) \in U)) \/
( (comUpdate(u1) \in U) /\ (preUpdate(u2) \in U))
NewInt4 ==  (u1 \in U /\ u2 \in U /\ u1.st="activated"
/\ u2.st="requested") ~>
((u1 \in U /\ u2 \in U /\ u1.st="completed" /\ u2.st="activated") \/
(u1 \in U /\ u2 \in U /\ u1.st="terminated" /\ u2.st="activated"))
=============================================================================
\* Modification History
\* Last modified 2014 by groban
\* Created 2014 by groban
```

Policy example with revoke usage method 2

```
------------------------------ MODULE Policy2 ------------------------------

EXTENDS Integers, FiniteSets, TLC, Reals


VARIABLES U

\* DECLARATION OF CONSTANT VALUES FOR S,A,O

\* (***************************************************************************

s1== [ id |-> 1, att |-> 10]

s2== [ id |-> 2, att |-> 20]

s3== [ id |-> 3, att |-> 30]

s4== [ id |-> 4, att |-> 40]

s5== [ id |-> 5, att |-> 50]


a1== [ id |-> 1, att |-> 11]

a2== [ id |-> 2, att |-> 21]

a3== [ id |-> 3, att |-> 31]

a4== [ id |-> 4, att |-> 41]

a5== [ id |-> 5, att |-> 51]


o1== [ id |-> 1, att |-> 12]

o2== [ id |-> 2, att |-> 22]

o3== [ id |-> 3, att |-> 32]

o4== [ id |-> 4, att |-> 42]

o5== [ id |-> 5, att |-> 52]
```

```
\* THE SETS OF ENTITIES
S == {s1,s2,s3}
A == {a1}
O == {o1,o2,o3}

----------------------------------------------------------------------------

\* POLICY RULES (WITH LEAST ACTIVATED USES)
\* (************************************************************************
SelectSubject(x) == CHOOSE s \in S : s.id=x.sid

UsesOfSubject(x) == {a \in U: a.sid=x.id /\ a.st="activated"}
NumOfUses(x) == Cardinality(UsesOfSubject(x))
Predicate(x,y) ==   /\ NumOfUses(SelectSubject(x))#1
                    /\ (NumOfUses(SelectSubject(x))>NumOfUses(SelectSubject(y)))

SameObjectAct(x,y) == x.oid = y.oid /\ x.st="activated"
ActivatedOnObject(x) == {u \in U: SameObjectAct(u,x)}
NumOfActOnObject(x) == Cardinality(ActivatedOnObject(x))

\* USE UPDATE ATTRIBUTES
\* (************************************************************************
createUse(x) == [sid|->x[1].id, aid|->x[2].id, oid|->x[3].id, st|->"requested",
att|->1]
preUpdate(x) == [x EXCEPT !.st="activated"]
onUpdate(x) == [x EXCEPT !.st="activated"]
```

```
denUpdate(x) == [x EXCEPT !.st="denied"]

termUpdate(x) == [x EXCEPT !.st="terminated"]

comUpdate(x) == [x EXCEPT !.st="completed"]



\*        POLICY IMPLEMENTATION

\* (***************************************************************************

Init == U = {}

vars == <<U>>


Request ==  \E u \in (S \X A \X O):

                /\ \forall x \in U : (x.sid#u[1].id \/ x.aid#u[2].id \/

 x.oid#u[3].id)

                /\ U' = U \cup {createUse(u)}


preEvaluate == \E u \in U:  (

                    /\ u.st="requested"

                    /\  \/  (   /\ NumOfActOnObject(u) = 0

                   /\ U' = (U \ {u}) \cup {preUpdate(u)}

)

                    \/  (   /\ NumOfActOnObject(u) = 1

                    /\ LET act==CHOOSE x \in U: SameObjectAct(x,u)

                    IN

  IF (Predicate(act,u))

                    THEN

    U' = (U \ {u}) \cup {termUpdate(u)}
```

```
                    ELSE
U' = (U \ {u,act}) \cup {preUpdate(u),termUpdate(act)}
                            )
                    )

Complete == \E u \in U:  (
                /\ u.st="activated"
                /\ U' = (U \ {u}) \cup {comUpdate(u)} )

Next == Request \/ preEvaluate \/ Complete

Spec == Init /\ [][Next]_vars /\ WF_vars(Next)

----------------------------------------------------------------------------

\* TYPECORRECTNESS
\* (*************************************************************************
USTATE == {"requested","activated","denied","terminated","completed"}
SubjectIDs == { s.id : s \in S }
ActionIDs == { a.id : a \in A }
ObjectIDs == { o.id : o \in O }
Entities == [ sid : SubjectIDs,
              aid : ActionIDs,
              oid : ObjectIDs,
              st : USTATE,
              att : {1} ]
```

```
TypeCorrectness == U \subseteq Entities

\* HIGH LEVEL PROPERTIES
\* (*************************************************************************
u1 == createUse(<<s1, a1, o1>>)
u2 == createUse(<<s2, a1, o1>>)

NewInt2 == \forall u \in U : NumOfActOnObject(u)<=1
NewInt3 ==  (preUpdate(u1) \in U /\ u2 \in U /\
Predicate(u2,u1)) ~> ( (termUpdate(u1) \in U) /\ (preUpdate(u2) \in U))
\/ ( (comUpdate(u1) \in U) /\ (preUpdate(u2) \in U))
NewInt4 == (u1 \in U /\ <> (u2 \in U) ) => u1 \in U

=============================================================================
\* Modification History
\* Last modified 2014 by groban
\* Created 2014 by groban
```

Appendix B

Publications

Referred papers in proceeding of international conferences and workshops

1. **A Use-based Approach for Enhancing UCON**

   (co-authors: Antonios Gouglidis, Ioannis Mavridis)

   Security and Trust Management (STM) 2013.

   Abstract: The security related characteristics of entities, the contextual information that describes them and the previous or concurrent usages exercised in the system are the criteria that the Usage CONtrol (UCON) family of models utilizes in the usage decision process. In this paper, a detailed classication of the aforementioned criteria along with a representative usage scenario for each category is presented, unveiling a number of UCON's limitations. In turn, a Use-based Usage CONtrol (UseCON) model is proposed that provides, for the creation of a usage decision, enhanced handling of information regarding context and previ- ous or current usages exercised in the system. The enhanced capabilities of the proposed approach are demonstrated and

discussed with the use of detailed application examples.

2. **Challenging Issues of UCON in Modern Computing Environments**

   (co-authors: Ioannis Mavridis)

   Proceedings of the fifth Balkan Conference in Informatics 2012.

   Abstract: Usage CONtrol (UCON) is a next generation access control model enhanced with capabilities presented in trust and digital rights management. However, modern computing environments are usually introducing complex usage scenarios. Such a complexity results in involving a large number of entities and in utilizing multi party contextual information during the decision making process of a particular usage. Moreover, usage control is demanded to support novel access modes on single or composite resources, while taking into account new socio-technical abstractions and relations. In this paper, a number of challenging issues faced when UCON is applied in modern computing environments are highlighted through the utilization of representative usage scenarios. The results of this study are revealing various limitations in contextual information handling, lack to support complicated usage modes of subjects on objects, and weaknesses in utilizing information concerning previous or current usages of system resources.

3. **Towards Use-Based Usage Control**

   (co-authors: Ioannis Mavridis)

   Proceedings of the 27th IFIP International Information Security and Privacy Conference (SEC 2012).

   Abstract: In this paper, a new Use-based usage CONtrol (UseCON) approach that supports recording of usages with the help of a new entity, named use, is presented. Uses provide information for the latest state (requested, active, denied, completed

or terminated) of every usage and facilitate the fine-grained definition and proper association of attributes to various system entities. The proposed approach provides enhanced contextual information modeling, support of complicated access modes and an alternative approach in obligations modeling. Moreover, UseCON is characterized by high expressiveness and ability to define policy rules in almost natural language.

4. **Towards Differentiated Utilization of Attribute Mutability for Access Control in Ubiquitous Computing**

   (co-authors: Ioannis Mavridis)

   Proceeding of the 14th Panhellenic Conference on Informatics (PCI 2010).

   Abstract: The operational characteristics of ubiquitous computing environments (UbiCom) generate new access control requirements which existing classical access control models fail to support efficiently. However, the Usage Control (UCON) family of models introduces components and mechanisms that seem to be able to partially match the specific requirements imposed by UbiCom environments. In this paper, an evaluation of current access control models based on a brief study of UbiCom access control requirements is presented. Then, a new access control approach that extends UCON towards a differentiated utilization of attribute mutability for easiness of administration, better performance and lower operational cost in UbiCom environments is proposed.

References

Tla+ tools, April 2013. URL http://research.microsoft.com/en-us/um/people/lamport/tla/tools.html.

The coq proof assistant, May 2014. URL http://coq.inria.fr/.

Tla+ toolbox user's guide, 2014. URL https://tla.msr-inria.inria.fr/tlatoolbox/doc/contents.html.

Gregory D. Abowd and Elizabeth D. Mynatt. Charting past, present, and future research in ubiquitous computing. *ACM Trans. Comput.-Hum. Interact.*, 7(1):29–58, March 2000. ISSN 1073-0516. doi: 10.1145/344949.344988. URL http://doi.acm.org/10.1145/344949.344988.

Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, HUC '99, pages 304–307, London, UK, UK, 1999. Springer-Verlag. ISBN 3-540-66550-1. URL http://dl.acm.org/citation.cfm?id=647985.743843.

Abulgader Almutairi and François Siewe. Ca-ucon: A context-aware usage control model. In *Proceedings of the 5th ACM International Workshop on Context-Awareness for Self-Managing Systems*, CASEMANS '11, pages 38–43, New York, NY, USA, 2011. ACM.

ISBN 978-1-4503-0877-9. doi: 10.1145/2036146.2036153. URL http://doi.acm.org/10.1145/2036146.2036153.

ANSI. ANSI INCITS 359-2004, role based access control, 2004.

Donovan Artz and Yolanda Gil. A survey of trust in computer science and the semantic web. *Web Semant.*, 5(2):58–71, June 2007. ISSN 1570-8268. doi: 10.1016/j.websem.2007.03.002. URL http://dx.doi.org/10.1016/j.websem.2007.03.002.

Kevin Ashton. That "internet of things" thing. *RFiD Journal*, 22:97–114, 2009.

Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

Jean Bacon, Ken Moody, and Walt Yao. A model of oasis role-based access control and its support for active security. *ACM Trans. Inf. Syst. Secur.*, 5(4):492–540, November 2002. ISSN 1094-9224. doi: 10.1145/581271.581276. URL http://doi.acm.org/10.1145/581271.581276.

Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008. ISBN 026202649X, 9780262026499.

Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.*, 2(4):263–277, June 2007. ISSN 1743-8225. doi: 10.1504/IJAHUC.2007.014070. URL http://dx.doi.org/10.1504/IJAHUC.2007.014070.

M. Ben-Ari. *Principles of Concurrent and Distributed Programming (2Nd Edition) (Prentice-Hall International Series in Computer Science)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006. ISBN 032131283X.

Messaoud Benantar. *Access Control Systems: Security, Identity Management and Trust Models*. Springer-Verlag New York, Inc., 2005.

Fran Berman, Geoffrey Fox, and Anthony J. G. Hey. *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons, Inc., New York, NY, USA, 2003. ISBN 0470853190.

Azzedine Boukerche and Yonglin Ren. A trust-based security system for ubiquitous and pervasive computing environments. *Comput. Commun.*, 31(18):4343–4351, December 2008. ISSN 0140-3664. doi: 10.1016/j.comcom.2008.05.007. URL `http://dx.doi.org/10.1016/j.comcom.2008.05.007`.

Sabrina Capitani di Vimercati, Sara Foresti, and Pierangela Samarati. Authorization and access control. In Milan Petkovic and Willem Jonker, editors, *Security, Privacy, and Trust in Modern Data Management*, Data-Centric Systems and Applications, pages 39–53. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-69861-6. URL `http://dx.doi.org/10.1007/978-3-540-69861-6_4`.

Guanling Chen, David Kotz, et al. A survey of context-aware mobile computing research. Technical report, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, 2000.

H.S. Cheng, Daqing Zhang, and J.G. Tan. Protection of privacy in pervasive computing environments. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, volume 2, pages 242–247 Vol. 2, April 2005. doi: 10.1109/ITCC.2005.233.

Michael J. Covington, Wende Long, Srividhya Srinivasan, Anind K. Dev, Mustaque Ahamad, and Gregory D. Abowd. Securing context-aware applications using environment roles. In *Proceedings of the Sixth ACM Symposium on Access Control Models and*

Technologies, SACMAT '01, pages 10–20, New York, NY, USA, 2001. ACM. ISBN 1-58113-350-2. doi: 10.1145/373256.373258. URL http://doi.acm.org/10.1145/373256.373258.

Jason Crampton and George Loizou. Administrative scope and role hierarchy operations. In *In Proceedings of Seventh ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, pages 145–154, 2002.

Frédéric Cuppens and Nora Cuppens-Boulahia. Modeling contextual security policies. *International Journal of Information Security*, 7(4):285–305, 2008. ISSN 1615-5262. doi: 10.1007/s10207-007-0051-9. URL http://dx.doi.org/10.1007/s10207-007-0051-9.

Ernesto Damiani, Sabrina De Capitani di Vimercati, and Pierangela Samarati. New paradigms for access control in open environments. In *SIGNAL PROCESSING AND INFORMATION TECHNOLOGY*, pages 540–545, 2005.

Pari Delir Haghighi, Shonali Krishnaswamy, Arkady Zaslavsky, and MohamedMedhat Gaber. Reasoning about context in uncertain pervasive computing environments. In Daniel Roggen, Clemens Lombriser, Gerhard Tröster, Gerd Kortuem, and Paul Havinga, editors, *Smart Sensing and Context*, volume 5279 of *Lecture Notes in Computer Science*, pages 112–125. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-88792-8. doi: 10.1007/978-3-540-88793-5_9. URL http://dx.doi.org/10.1007/978-3-540-88793-5_9.

Stelios Dritsas, Dimitris Gritzalis, and Costas Lambrinoudakis. Protecting privacy and anonymity in pervasive computing: Trends and perspectives. *Telemat. Inf.*, 23(3): 196–210, August 2006. ISSN 0736-5853. doi: 10.1016/j.tele.2005.07.005. URL http://dx.doi.org/10.1016/j.tele.2005.07.005.

Yitao Duan and John Canny. Protecting user data in ubiquitous computing: Towards trustworthy environments. In David Martin and Andrei Serjantov, editors, *Privacy Enhancing Technologies*, volume 3424 of *Lecture Notes in Computer Science*, pages 167–185. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-26203-9. doi: 10.1007/11423409_11. URL http://dx.doi.org/10.1007/11423409_11.

David F. Ferraiolo, D. Richard Kuhn, and Ramaswamy Chandramouli. *Role-Based Access Control*. Artech House, Inc., 2003.

I. Foster, Zhao Yong, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1–10, 2008.

Ian Foster and Carl Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003. ISBN 1558609334.

Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid - enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15:2001, 2001.

Christos K. Georgiadis, Ioannis Mavridis, George Pangalos, and Roshan K. Thomas. Flexible team-based access control using contexts. In *Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies*, SACMAT '01, pages 21–27, New York, NY, USA, 2001. ACM. ISBN 1-58113-350-2. doi: 10.1145/373256.373259. URL http://doi.acm.org/10.1145/373256.373259.

Dieter Gollmann. *Computer Security*. John Wiley & Sons, Inc., New York, NY, USA, 1999. ISBN 0-471-97844-2.

Antonios Gouglidis and Ioannis Mavridis. On the definition of access control require-

ments for grid and cloud computing systems. In Anastasios Doulamis, Joe Mambretti, Ioannis Tomkos, and Theodora Varvarigou, editors, *Networks for Grid Applications*, volume 25 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 19–26. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-11732-9. doi: 10.1007/978-3-642-11733-6_3. URL http://dx.doi.org/10.1007/978-3-642-11733-6_3.

Antonios Gouglidis, Ioannis Mavridis, and VincentC. Hu. Security policy verification for multi-domains in cloud systems. *International Journal of Information Security*, pages 1–15, 2013. ISSN 1615-5262. doi: 10.1007/s10207-013-0205-x. URL http://dx.doi.org/10.1007/s10207-013-0205-x.

Dimitris Gritzalis, M Kandias, V Stavrou, and L Mitrou. History of information: The case of privacy and security in social media. *Proc. of the History of Information Conference*, 2014.

Christos Grompanopoulos and Ioannis Mavridis. Towards differentiated utilization of attribute mutability for access control in ubiquitous computing. *Informatics, Panhellenic Conference on*, 0:118–123, 2010. doi: http://doi.ieeecomputersociety.org/10.1109/PCI.2010.45.

Christos Grompanopoulos and Ioannis Mavridis. Towards use-based usage control. In Dimitris Gritzalis, Steven Furnell, and Marianthi Theoharidou, editors, *Information Security and Privacy Research*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 585–590. Springer Boston, 2012a. ISBN 978-3-642-30435-4.

Christos Grompanopoulos and Ioannis Mavridis. Challenging issues of ucon in modern computing environments. In *Proceedings of the Fifth Balkan Conference in Informatics*,

BCI '12, pages 156–161, New York, NY, USA, 2012b. ACM. ISBN 978-1-4503-1240-0. doi: 10.1145/2371316.2371346. URL http://doi.acm.org/10.1145/2371316.2371346.

Christos Grompanopoulos, Antonios Gouglidis, and Ioannis Mavridis. A use-based approach for enhancing ucon. In *Security and Trust Management*, pages 81–96. Springer Berlin Heidelberg, 2013.

Jinsong Gui, Zhigang Chen, and Xiaoheng Deng. An improved ucona-based authorization policy specification for ubiquitous systems. In Yuanxu Yu, Zhengtao Yu, and Jingying Zhao, editors, *Computer Science for Environmental Engineering and EcoInformatics*, volume 158 of *Communications in Computer and Information Science*, pages 151–158. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-22693-9. doi: 10.1007/978-3-642-22694-6_21. URL http://dx.doi.org/10.1007/978-3-642-22694-6_21.

Keijo Heljanko. Model checking based software verification, 2006. URL http://iplu.vtt.fi/digitalo/modelchecking.pdf.

Chung Tong Hu, David F Ferraiolo, David R Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone. Guide to attribute based access control (abac) definition and considerations. NIST SP - 800-162, 2014.

Vincent C Hu, David Ferraiolo, and D Richard Kuhn. *Assessment of access control systems*. US Department of Commerce, National Institute of Standards and Technology, 2006.

Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press, New York, NY, USA, 2004. ISBN 052154310X.

ITU-T. X.812 recommendation, 1995.

Helge Janicke, Antonio Cau, and Hussein Zedan. A note on the formalisation of ucon. In *Proceedings of the 12th ACM symposium on Access control models and technologies*,

SACMAT '07, pages 163–168, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-745-2.

Xin Jin, Ram Krishnan, and Ravi Sandhu. A unified attribute-based access control model covering dac, mac and rbac. In Nora Cuppens-Boulahia, Frédéric Cuppens, and Joaquin Garcia-Alfaro, editors, *Data and Applications Security and Privacy XXVI*, volume 7371 of *Lecture Notes in Computer Science*, pages 41–55. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-31539-8. doi: 10.1007/978-3-642-31540-4_4. URL http://dx.doi.org/10.1007/978-3-642-31540-4_4.

A Joshi, T Finin, L Kagal, J Parker, and A Patwardhan. Security policies and trust in ubiquitous computing. *Philos Trans A Math Phys Eng Sci*, 366(1881):3769–3780–, October 2008. URL http://europepmc.org/abstract/MED/18672450.

L. Kagal, T. Finin, and A. Joshi. Trust-based security in pervasive computing environments. *Computer*, 34(12):154–157, Dec 2001. ISSN 0018-9162. doi: 10.1109/2.970591.

L. Kagal, T. Finin, Anupam Joshi, and S. Greenspan. Security and privacy challenges in open and dynamic environments. *Computer*, 39(6):89 –91, june 2006. ISSN 0018-9162. doi: 10.1109/MC.2006.207.

Lalana Kagal, Tim Finin, and Anupam Joshi. A policy language for a pervasive computing environment. In *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, POLICY '03, pages 63–, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1933-4. URL http://dl.acm.org/citation.cfm?id=826036.826875.

Ekkart Kindler. Safety and liveness properties: A survey. *Bulletin of the European Association for Theoretical Computer Science*, 53:268–272, 1994.

Leanid Krautsevich, Aliaksandr Lazouski, Fabio Martinelli, Paolo Mori, and Artsiom Yautsiukhin. Usage control, risk and trust. In Sokratis Katsikas, Javier Lopez, and Miguel Soriano, editors, *Trust, Privacy and Security in Digital Business*, volume 6264 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15151-4. doi: 10.1007/978-3-642-15152-1_1. URL http://dx.doi.org/10.1007/978-3-642-15152-1_1.

Devdatta Kulkarni and Anand Tripathi. Context-aware role-based access control in pervasive computing systems. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, SACMAT '08, pages 113–122, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-129-3. doi: 10.1145/1377836.1377854. URL http://doi.acm.org/10.1145/1377836.1377854.

Arun Kumar, Neeran Karnik, and Girish Chafle. Context sensitivity in role-based access control. *ACM SIGOPS Operating Systems Review*, 36(3):53–66, 2002.

Leslie Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002. ISBN 032114306X.

Leslie Lamport, John Matthews, Mark Tuttle, and Yuan Yu. Specifying and verifying systems with tla+. In *Proceedings of the 10th workshop on ACM SIGOPS European workshop*, EW 10, pages 45–48, New York, NY, USA, 2002. ACM. doi: 10.1145/1133373.1133382. URL http://doi.acm.org/10.1145/1133373.1133382.

Butler W. Lampson. Protection. *SIGOPS Oper. Syst. Rev.*, 8:18–24, January 1974. ISSN 0163-5980.

Bo Lang, Ian Foster, Frank Siebenlist, Rachana Ananthakrishnan, and Tim Freeman. A flexible attribute based access control method for grid computing. *Journal of Grid Com-*

puting, 7(2):169–180, 2009. ISSN 1570-7873. doi: 10.1007/s10723-008-9112-1. URL http://dx.doi.org/10.1007/s10723-008-9112-1.

Aliaksandr Lazouski, Fabio Martinelli, and Paolo Mori. Usage control in computer security: A survey. *Computer Science Review*, 4(2):81 – 99, 2010. ISSN 1574-0137.

Ninghui Li, Ji-Won Byun, and Elisa Bertino. A critique of the ANSI standard on role-based access control. *IEEE Security and Privacy*, 5(6):41–49, 2007.

Jianfeng Lu, Ruixuan Li, Jinwei Hu, and Dewu Xu. Static enforcement of static separation-of-duty policies in usage control authorization models. *IEICE Transactions*, 95-B(5): 1508–1518, 2012.

Fabio Martinelli and Paolo Mori. On usage control for grid systems. *Future Gener. Comput. Syst.*, 26(7):1032–1042, July 2010. ISSN 0167-739X. doi: 10.1016/j.future.2009.12.005. URL http://dx.doi.org/10.1016/j.future.2009.12.005.

Patrick McDaniel. On context in authorization policy. In *Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies*, SACMAT '03, pages 80–89, New York, NY, USA, 2003. ACM. ISBN 1-58113-681-1. doi: 10.1145/775412.775422. URL http://doi.acm.org/10.1145/775412.775422.

Kazuhiro Minami and David Kotz. Secure context-sensitive authorization. In *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, PERCOM '05, pages 257–268, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2299-8. doi: 10.1109/PERCOM.2005.37. URL http://dx.doi.org/10.1109/PERCOM.2005.37.

Asmund Ahlmann Nyre. Usage control enforcement - a survey. In *Proceedings of the IFIP WG 8.4/8.9 International Cross Domain Conference on Availability, Reliability and*

Security for Business, Enterprise and Health Information Systems, ARES'11, pages 38–49, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-23299-2. URL http://dl.acm.org/citation.cfm?id=2033973.2033978.

OASIS. Oasis extensible access control markup language (xacml) tc, 2011. URL http://www.oasis-open.org/.

Sejong Oh and Ravi Sandhu. A model for role administration using organization structure. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 155–162. ACM, 2002.

Jaehong Park and Ravi Sandhu. The ucon abc usage control model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174, 2004.

A. Pretschner, M. Hilty, F. Schutz, C. Schaefer, and T. Walter. Usage control enforcement: Present and future. *Security Privacy, IEEE*, 6(4):44–53, July 2008. ISSN 1540-7993. doi: 10.1109/MSP.2008.101.

A. Pretschner, J. Ruesch, C. Schaefer, and T. Walter. Formal analyses of usage control policies. In *Availability, Reliability and Security, 2009. ARES '09. International Conference on*, pages 98–105, 2009. doi: 10.1109/ARES.2009.100.

Lili Qiu, Yin Zhang, Feng Wang, Mi Kyung, and Han Ratul Mahajan. Trusted computer system evaluation criteria. In *National Computer Security Center*, 1985.

P.V. Rajkumar, S.K. Ghosh, and P. Dasgupta. Concurrent usage control implementation verification using the spin model checker. In Natarajan Meghanathan, Selma Boumerdassi, Nabendu Chaki, and Dhinaharan Nagamalai, editors, *Recent Trends in Network Security and Applications*, volume 89 of *Communications in Computer and Information Science*, pages 214–223. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-

14477-6. doi: 10.1007/978-3-642-14478-3_22. URL http://dx.doi.org/10.1007/978-3-642-14478-3_22.

A. Ranganathan, J. Al-Muhtadi, and R.H. Campbell. Reasoning about uncertain contexts in pervasive computing environments. *Pervasive Computing, IEEE*, 3(2):62–70, April 2004. ISSN 1536-1268. doi: 10.1109/MPRV.2004.1316821.

Kumar Ranganathan. Trustworthy pervasive computing: The hard security problems. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, PERCOMW '04, pages 117–, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2106-1. URL http://dl.acm.org/citation.cfm?id=977405.978635.

Silvio Ranise and Alessandro Armando. On the automated analysis of safety in usage control: A new decidability result. In Li Xu, Elisa Bertino, and Yi Mu, editors, *Network and System Security*, volume 7645 of *Lecture Notes in Computer Science*, pages 15–28. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-34600-2. doi: 10.1007/978-3-642-34601-9_2. URL http://dx.doi.org/10.1007/978-3-642-34601-9_2.

Debashis Saha and Amitava Mukherjee. Pervasive computing: A paradigm for the 21st century. *Computer*, 36(3):25–31, March 2003. ISSN 0018-9162. doi: 10.1109/MC.2003.1185214. URL http://dx.doi.org/10.1109/MC.2003.1185214.

Pierangela Samarati and Sabrina De Capitani di Vimercati. Access control: Policies, models, and mechanisms. In *Revised versions of lectures given during the IFIP WG 1.7 International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design: Tutorial Lectures*, FOSAD '00, pages 137–196, London, UK, UK, 2001a. Springer-Verlag. ISBN 3-540-42896-8.

Pierangela Samarati and SabrinaCapitani Vimercati. Access control: Policies, models, and mechanisms. In Riccardo Focardi and Roberto Gorrieri, editors, *Foundations of Security Analysis and Design*, volume 2171 of *Lecture Notes in Computer Science*, pages 137–196. Springer Berlin Heidelberg, 2001b. ISBN 978-3-540-42896-1. doi: 10.1007/3-540-45608-2_3. URL http://dx.doi.org/10.1007/3-540-45608-2_3.

R. Sandhu. The ascaa principles for access control interpreted for collaboration systems. In *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*, pages 532–532, May 2008. doi: 10.1109/CTS.2008.4543974.

Ravi Sandhu and Jaehong Park. Usage control: A vision for next generation access control. In *Computer Network Security*, volume 2776, pages 17–31. Springer Berlin / Heidelberg, 2003.

Ravi Sandhu and Pierangela Samarati. Authentication, access control, and audit. *ACM Comput. Surv.*, 28(1):241–243, March 1996. ISSN 0360-0300. doi: 10.1145/234313.234412. URL http://doi.acm.org/10.1145/234313.234412.

Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer. The arbac97 model for role-based administration of roles. *ACM Trans. Inf. Syst. Secur.*, 2(1):105–135, 1999.

Ravi S. Sandhu. Lattice-based access control models, 1993.

R.S. Sandhu and P. Samarati. Access control: principle and practice. *Communications Magazine, IEEE*, 32(9):40–48, Sept 1994. ISSN 0163-6804. doi: 10.1109/35.312842.

R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *Computer*, 29(2):38 –47, feb 1996. ISSN 0018-9162. doi: 10.1109/2.485845.

Marianthi Theoharidou, Nick Papanikolaou, Siani Pearson, and Dimitris Gritzalis. Privacy risk, security, accountability in the cloud. In *Proceedings of the 2013 IEEE International*

Conference on Cloud Computing Technology and Science - Volume 01, CLOUDCOM '13, pages 177–184, Washington, DC, USA, 2013. IEEE Computer Society. ISBN 978-0-7695-5095-4. doi: 10.1109/CloudCom.2013.31. URL http://dx.doi.org/10.1109/CloudCom.2013.31.

R.K. Thomas and R. Sandhu. Models, protocols, and architectures for secure pervasive computing: challenges and research directions. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 164–168, March 2004. doi: 10.1109/PERCOMW.2004.1276925.

Alessandra Toninelli, R. Montanari, L. Kagal, and O. Lassila. Proteus: A semantic context-aware adaptive policy model. In *Policies for Distributed Systems and Networks, 2007. POLICY '07. Eighth IEEE International Workshop on*, pages 129–140, June 2007. doi: 10.1109/POLICY.2007.40.

Lingyu Wang, Duminda Wijesekera, and Sushil Jajodia. A logic-based framework for attribute based access control. In *Proceedings of the 2004 ACM Workshop on Formal Methods in Security Engineering*, FMSE '04, pages 45–55, New York, NY, USA, 2004. ACM. ISBN 1-58113-971-3. doi: 10.1145/1029133.1029140. URL http://doi.acm.org/10.1145/1029133.1029140.

Mark Weiser. Ubiquitous computing. In *ACM Conference on Computer Science*, page 418, 1994.

Mark Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11, July 1999. ISSN 1559-1662. doi: 10.1145/329124.329126. URL http://doi.acm.org/10.1145/329124.329126.

Jeannette M. Wing. A specifier's introduction to formal methods. *Computer*, 23(9):8–23,

September 1990. ISSN 0018-9162. doi: 10.1109/2.58215. URL http://dx.doi.org/10.1109/2.58215.

Ran Yang, Chuang Lin, and Fujun Feng. A time and mutable attribute-based access control model. *JCP*, 4(6):510–518, 2009. URL http://dblp.uni-trier.de/db/journals/jcp/jcp4.html#YangLF09.

Eric Yuan and Jin Tong. Attributed based access control (abac) for web services, 2005.

G. Zhang and M. Parashar. Dynamic context-aware access control for grid applications. In *Grid Computing, 2003. Proceedings. Fourth International Workshop on*, pages 101–108. IEEE, 2004. ISBN 076952026X.

Xinwen Zhang. *Formal Model and Analysis of Usage Control*. PhD thesis, Fairfax, VA, USA, 2006. AAI3221391.

Xinwen Zhang, Francesco Parisi-Presicce, Ravi Sandhu, and Jaehong Park. Formal model and policy specification of usage control. *ACM Trans. Inf. Syst. Secur.*, 8:351–387, November 2005. ISSN 1094-9224.

Xinwen Zhang, Ravi Sandhu, and Francesco Parisi-Presicce. Safety analysis of usage control authorization models. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, ASIACCS '06, pages 243–254, New York, NY, USA, 2006. ACM. ISBN 1-59593-272-0.

Xinwen Zhang, Masayuki Nakae, Michael J. Covington, and Ravi Sandhu. Toward a usage-based security framework for collaborative computing systems. *ACM Trans. Inf. Syst. Secur.*, 11(1):1–36, 2008.