Towards an Authorization Model for Distributed Systems based on the Semantic Web

Jose M. Alcaraz Calero[a,b], Gregorio Martinez Perez[b,*], Antonio F. Gomez Skarmeta[b]

[a]*Automated Infrastructure Lab*
Hewlett Packard Laboratories Bristol
Filton Rd, Stroke Gifford
BS34 8QZ Bristol, UK
[b]*Department of Communications and Information Engineering*
University of Murcia
Computer Science Faculty, Espinardo
30011, Murcia Spain

Abstract

Authorization is a crucial process in current information systems. Nowadays, many of the current authorization systems do not provide methods to describe the semantics of the underlying information model which they are protecting. This fact can lead to mismatch problems between the semantics of the authorization model and the semantics of the underlying data and resources being protected. In order to solve this problem, this paper describes an authorization model based on Semantic Web technologies. This authorization model uses the Common Information Model (CIM) to represent the underlying information model. For this reason, a new conversion process of CIM into the Semantic Web languages has been proposed converting properly the semantics available in the CIM model. This representation provides a suitable information model based on a well-known logic formalism for implementing the authorization model and a formal language for describing concisely the semantic of the information models being protected. The formal authorization model supports role-based access control (RBAC), hierarchical RBAC, conditional RBAC and object hierarchies, among other features. Moreover, this paper describes an authorization architecture for distributed systems taking into account aspects such as privacy among parties and trust management. Finally, some implementation aspects of this system have been also described.

Key words: Authorization Model, Distributed System, Semantic Web, Authorization Architecture, Formal Methods

1. Introduction

Nowadays, there exist many authorization systems for distributed environments. There are authorization systems for Multi-agent systems [1], GRID structures [2], P2P systems [3], federated scenarios [4], Cloud Computing [5] and in general terms, any relevant technology associated to distributed architectures. However, some of the current

*Corresponding Author
Email addresses: jmalcaraz@um.es (Jose M. Alcaraz Calero), gregorio@um.es (Gregorio Martinez Perez), skarmeta@um.es (Antonio F. Gomez Skarmeta)

authorization systems do not provide methods to describe the semantics of the underlying information model which they are protecting. This fact can lead to mismatch problems between the semantics of the authorization model and the semantics of the underlying data and resources being protected. For example, to express an authorization for a file path, the semantics associated to the file system structure have to be coded or declared. Analogously, this happens in distributed systems when the resource can be a whole computer and the semantics require that all the logical and physical services are affected by this decision. Thus, when any of these semantics is not implemented in the authorization system, this system lacks enough information for managing this kind of resource correctly.

Hence, the main aim of this paper is to foster an approach based on the usage of the Semantic Web technologies to provide the description of information systems in an accurate, formal and highly expressive manner and to describe a formal authorization model suitable for this description. This authorization model uses the Common Information Model (CIM) to represent the underlying information model. For this reason, a new conversion process of CIM into the Semantic Web languages has been proposed converting properly the semantics available in the CIM model. This representation provides a suitable information model based on a well-known logic formalism for implementing the authorization model and a formal language for describing concisely the semantic of the information models being protected. The idea behind this approach is to match the semantics of the authorization framework and the semantics of the underlying data being protected in order to avoid mismatch problems related to the semantics of these protected objects. The formal authorization model supports role-based access control (RBAC), hierarchical RBAC, conditional RBAC and object hierarchies, among other features. An authorization architecture for distributed systems is also tackled in this paper in order to validate the authorization model proposed taking into account aspects such as privacy among parties and trust management.

In order to explain the authorization model promoted, the paper has been structured as follows: section 2 describes the languages and logic formalisms used as foundation for the authorization model. The information model used for representing the information system and its conversion to the Semantic Web languages are described in section 3. Section 4 describes the underlying authorization model. After that, the authorization architecture for distributed environment is explained in section 5. Section 6 provides some implementation aspects related to the authorization systems. Finally, some conclusions and future works are provided in section 7.

2. Languages and Logic Formalism

The RDF [6], RDFS [7], OWL 2 [8] and SWRL [9] languages designed for the Semantic Web [10] are formal languages for representing knowledge. While RDF is intended as a general-propose language for describing resources in the Web, RDFS, OWL 2 and SWRL are general-purpose languages for describing semantics associated to these resources. These languages a have common basic well-known logic formalism called Description Logic [11]. This formalism provides a highly expressive language for describing conditional knowledge (SWRL), simple semantic features such as inheritance among concepts and properties (RDFS) and advanced semantic features such as transi-

tiveness and reflexiveness in properties or disjointness of concepts (OWL 2). At same time, the formalism is kept within decidability bounds, so that the inference processes are performed in a finite time. This set of languages and inference processes provides a suitable framework for defining an authorization model and an information model using semantic associated to them. Notice that, Semantic Web technologies [8], [9] allow any element to be extended in the architecture including the authorization model, the information model and the semantics of these, since they provide languages to express knowledge and semantics.

There is some basic notation related to these languages that will be used in the rest of this paper. RDF/OWL 2 manages concepts (or classes) and properties (or attributes). Thus, if A is a concept then $A.p$ denotes the property p belonging to the concept A. In the same way, $A \sqsubset B$ defines A as subclass of B and $A.p \sqsubset B.q$ defines p as subproperty of q. On the other hand, an SWRL rule is an implication relationship between the antecedent terms and consequent terms and its semantics define that if all the antecedent terms hold then all the consequent terms also should hold. Thus, the term $C(?x)$ represents any instance x of the concept C, the term $p(?x, ?y)$ represents a property p belonging to the instance x and having the value y, and the term $f : a(?x, ?y, .., ?z)$ represents the function a having as paramaters $x, y, ...z$. These built-in functions are used to carry out mathematical, logical and string calculations and the variables $?x, ?y, ?z$ will be bound to the instances and values available in the domain. Additionally, there are another two special terms denoted as $sameAs(?x, ?y)$ and $DifferentFrom(?x, ?y)$ for comparing the equality and difference of the variables $?x, ?y$. In order to clarify the SWRL notation, let's suppose the following rule: $File(?x) \wedge name(?x, ?y) \wedge f : endWith(?y, "/") \rightarrow Directory(?x)$. This rule expresses that for all instances x of the domain, if x is a $File$, this file has a name y, and this name ends with "/", then the instance x is also a $Directory$.

3. Information Model

There are several industrial standards for modelling information systems like CIM [12] and OIM [13]. In fact, CIM is being used in a wide variety of large systems such as SAP, Windows XP/2000 and VMWare. It has been chosen in our paper as base model for the representation of the information system. This model allows the representation of almost all the physically and logically elements available in a computer including devices, services, databases, applications, networks, security, users, policies, events and etcetera. CIM provides a wide variety of *concepts* and *associations* to represent the information systems. In particular, CIM is composed of more than 1600 concepts and associations in its latest version.

Figure 1 shows a brief summary of some important CIM concepts and their hierarchical structures. Each node in a branch represents an inheritance relationship to its parent. Thus, for example, *Role* can be interpreted as $Role \sqsubset Collection \sqsubset ManagedElement \sqsubset Thing$, and so on. In the context, $Thing$ is a super class that all the others classes inherit, implicitly. Each of these concepts have associated properties but for simplicity they are not shown.

Regarding CIM hierarchical structure, all the concepts managed are under the $ManagedElement$ branch whereas the associations are represented under the $Association$ branch. In the $ManagedElement$ branch there are useful

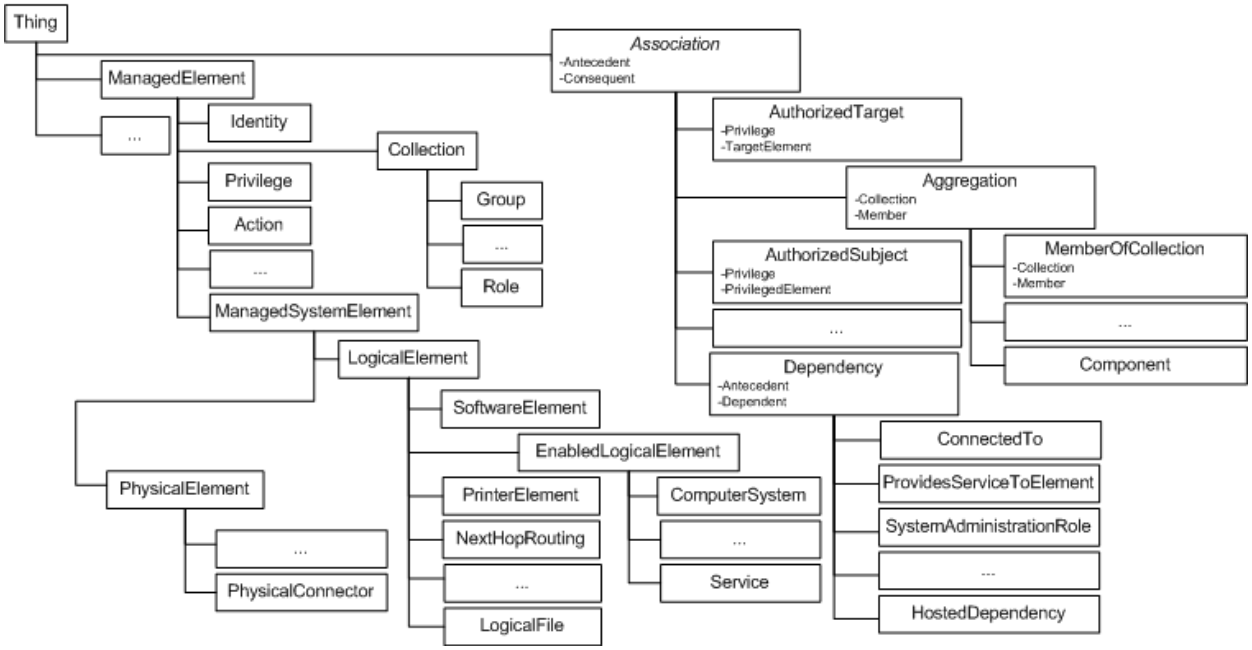

Figure 1: CIM hierarchy created in RDF/OWL 2 language

concepts for authorization purposes like *Identity*, *Privilege*, *Group* and *Role* lately described in section 4. There are also all the concepts related to the *ManagedSystemElement* which in turn, represents the logical and physical description of the elements present in the information system like *Printer*, *Software*, *Services* and *Files*. A notable feature of this structure is raised when an instance i belongs to any concept, for example *PrinterElement*, denoted as *PrinterElement*(i). Then as a result, the inference processes of the Description Logic will apply the *inheritance* semantics inferring that i is also an instance of *LogicalElement*, *ManagedSystemElement*, *ManagedElement* and *Thing* denoted as *LogicalElement*(i), *ManagedSystemElement*(i), *ManagedElement*(i) and *Thing*(i), respectively. Notice the added-value provided by all the information automatically generated by the underlying model from only a simple instance.

CIM is usually distributed in MOF and XML syntaxes. Thus, a conversion from MOF/XML to RDF/OWL 2 might be done in order to get a model suitable for our purposes. Actually, some research works [14], [15] have focused on carrying out this conversion process. This paper provides a new conversion process based on this earlier research. The main contribution of this new process resides in the new method for converting the CIM associations. While previous works map these CIM associations to OWL properties, this paper promotes the mapping of CIM association to OWL concepts. In fact, figure 1 represents a summary of the real RDF/OWL 2 hierarchy achieved. Notice that there is an *Association* branch not present in the previous works which is part of the original CIM. The previous works model this branch as properties of other classes. This new conversion process contains all the associations as concepts which in turn, due to their nature always contain at least two properties in order to spec-

ify the two different sides of the association (many of these properties have not been shown for simplicity). This branch is important in this paper for enabling the differentiation of the different kinds of associations available in the model. This cannot be done using previous models. In fact, it is worth mentioning the inheritance available among the properties available in the CIM associations. Thus, any property belonging to an association inherits the properties of the parent of this association. For example, just following the tree structure depicted in figure 1, the property $MemberOfCollection.Collection$ could be defined as $MemberOfCollection.Collection \sqsubset Aggregation.Collection \sqsubset Association.Antecedent$ and analogously, $MemberOfCollection.Member$ could be defined as $MemberOfCollection.Member \sqsubset Aggregation.Member \sqsubset Association.Consequent$, and so on. By applying these semantics, the inference processes of the Description Logic will infer from an instance i belonging to the concept the $MemberOfCollection$, not only that this instance is also belonging to $Aggregation$, $Associations$ and $Thing$ concepts but also that $MemberOfCollection.Member$ and $MemberOfCollection.Collection$ properties are also instances of the properties $Aggregation.Member$, $Association.Antecedent$ and $Aggregation.Collection$, $Association.Consequent$, respectively.

Notice that most of the current CIM provider implementations (software that provides CIM representation of the current state of the managed system) such as OpenPegasus, OpenWBEM, WBEMService and several commercial solutions do not provide inference processes in the CIM representation. However, the added value provided by the Description Logic inference process can be applied by doing the conversion of the CIM instances from XML to RDF/OWL 2.

4. Authorization Model

This section provides the formal model to control the access to resources in a distributed system. This model is based in the information model previously described in section 3. Figure 2 represents all the concepts of this information model related to the authorization model. Thus, in essence, an authorization model has to determine if a *subject* has the *privilege* to perform a given action over the controlled *object*. This fact can be represented by a 3-tuple $(Subject, Privilege, Object)$. Thus, these concepts can be directly associated with the information model depicted in figure 1, $Subject$ being directly mapped to the *Identity* CIM concept and *Privilege* to the *Privilege* CIM concept. Any CIM concept inheriting from *ManagedElement* can be mapped to an $Object$ and this enables access control of any concept represented in CIM. Moreover, each privilege asserted in the system, i.e. each instance of this 3-tuple, is represented in CIM by means of the instantiation of two different associations, $AuthorizationSubject$ and $AuthorizationTarget$. These associations represent the authorization relationship between ($Subject$ and *Privilege*) and (*Privilege* and $Object$), respectively.

Then, when the authorization system has to prove if a given 3-tuple has access or not, the system will try to find the authorization proof in the underlying knowledge base. This search for a formal proof can be done with the $SPARQL$ language [16], a language of the Semantic Web intended for this purpose. Essentially, the code depicted in figure 3 represents the query in $SPARQL$ syntax, an SQL-like syntax and this query can be used to find the authorization

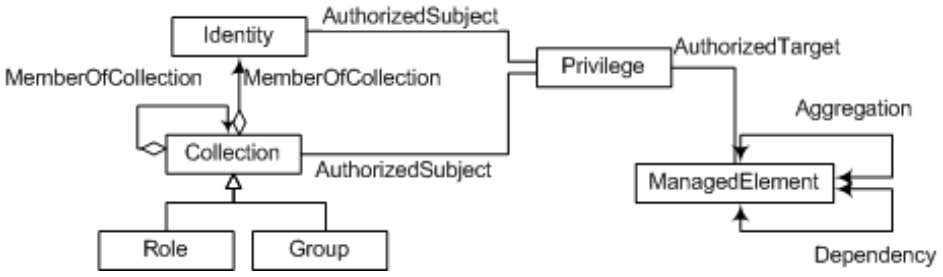

Figure 2: CIM concepts and associations related to the authorization model fostered

proof. This query receives as parameter the 3-tuple to check, then tries to find the evidence of the authorization proof by means of the criteria specified in the *WHERE* clause. This clause is used to search for the proof of authorization checking the existence of both associations *AuthorizationSubject* and *AuthorizationTarget* applied over the 3-tuple specified as parameter. Actually, lines 3-4 search for any identity ?*s* where a privilege ?*p* is assigned through the association ?*ac*, where ?*p* and ?*ac* are the name of the identity and the privilege specified as parameters. Lines 5-6 search for any managed element ?*o* associated to the previous privilege ?*p* through the association ?*at* where *instanceID* property of this element matches with the parameter provided. Notice that the authorization modelis based in a discretionary access control and therefore a *deny by default* policy can be applied. As a result of execution of this *SPARQL* query, if there is not evidence of authorization (0 rows returned), the request will be denied (deny by default) and otherwise it will be accepted.

1 $PARAMS(subject, privilege, object)$

2 $SELECT(?s, ?p, ?o, ?ac, ?at)$

3 $WHERE\ Identity(?s) \wedge name(?s, \$subject) \wedge Privilege(?p) \wedge activity(?p, \$privilege) \wedge privilegeGranted(?p, \text{"}true\text{"})$

4 $\quad AuthorizationSubject(?ac) \wedge privilege(?ac, ?p) \wedge privilegedElement(?ac, ?s) \wedge$

5 $\quad ManagedElement(?o) \wedge instanceID(?o, \$object) \wedge$

6 $\quad AuthorizationTarget(?at) \wedge privilege(?at, ?p) \wedge targetElement(?at, ?o)$

Figure 3: *SPARQL* query to provide the authorization proof

To support the authorization model a role-based access control (RBAC) [17] is incorporated. The authorization model relies on the usage of *roles* as a set of privileges that could be assigned to a subject, and *groups* as a set of users. The CIM association *MemberOfCollection* is used to assert that any identity is belonging to a collection, either *Role* or *Group* concepts. Moreover, a *Privilege* can be associated to either *Identity*, *Group* or *Role* by means of the previously mentioned *AuthorizationSubject* association. As a result, the privileges granted to either *Group* or *Role* have to be applied to the users belonging to these collections. This is a transitiveness relationship between *Group*/*Role* and *Identity* extended by means of the *AuthorizationSubject* associations. This transitiveness can be incorporated by

means of the inclusion of the semantic rule depicted in figure 4 to the authorization model.

1 $Collection(?c) \wedge AuthorizedSubject(?as) \wedge privilegedElement(?as, ?c) \wedge privilege(?as, ?p)$

2 $Identity(?i) \wedge MemberOfCollection(?mc) \wedge collection(?mc, ?c) \wedge member(?mc, ?i)$

3 $\rightarrow$

4 $AuthorizedSubject(?newas) \wedge privilege(?newas, ?p) \wedge privilegedElement(?newas, ?i)$

Figure 4: *SWRL* rule for supporting role-based access control

Lines 1-2 of the rule of the figure 4, it retrieves any privilege association $?as$ applied over a collection $?c$ (role or group) and any identity $?i$ belonging to this collection (by means of the $?mc$ association) and produce the transitiveness of the privilege creating a new grant directly applied to the identities $?i$. Notice that the association *MemberOf-Collection* have two sides: property *member* denotes the object which is part of the collection defined in the property *collection*. Then, the authorization proof previously described in figure 3 can also manage these privileges.

Additionally, hierarchical role based access control (hRBAC) [17] enables a hierarchical subject grouping and it has been incorporated to the authorization model. This kind of access control allows the definition of hierarchies of collections, roles and groups. A collection can be defined as a specialization of another more generic collection. For example, the role *DatabaseAdmin* could be defined as a specification of the *Admin* role, and therefore, all the privileges defined over *Admin* also apply for *DatabaseAdmin*. This transitiveness among collections could be defined by the rule given in figure 5.

1 $Collection(?ca) \wedge AuthorizedSubject(?as) \wedge privilegedElement(?as, ?ca) \wedge privilege(?as, ?p)$

2 $Collection(?cb) \wedge MemberOfCollection(?mc) \wedge collection(?mc, ?ca) \wedge member(?mc, ?cb)$

3 $\rightarrow$

4 $AuthorizedSubject(?newas) \wedge privilege(?newas, ?p) \wedge privilegedElement(?newas, ?cb)$

Figure 5: *SWRL* rule for supporting hierarchical role-based access control and hierarchical subjects grouping

Line 1 of the rule given in figure 5 retrieves any authorization association $?as$ between a collection $?ca$ and a privilege $?p$. Line 2 retrieves any collection $?cb$ belonging to the previously obtained collections $?ca$. As a result, the most specific collections $?cb$ receive all the privileges directly associated to their parent by the direct association of these privileges using the new association $?newas$. Thus, combining rules given in figures 4 and 5, subjects have assigned all the privileges directly and indirectly inferred, and the authorization proof given in the figure 3 is able to manage hRBAC.

To support an object hierarchy [18] provides a powerful expressiveness in the authorization model. This feature allows the scope of a privilege applied to a given object to be extended to other objects. For example, let's suppose a *FileSystem* concept is composed of a *Directory* which in turn, it is composed of a *File*. A privilege could be granted

to a *FileSystem*. The object hierarchy will enable to infer automatically the extension of the scope of this privilege to the elements composing the file system, i.e. *Directory* and *Files*. Analogously, a privilege at the level of *ComputerSystem* means that all the elements related to this computer receive this privilege. Notice the added value of this feature for administration tasks, in which there are formal methods to validate this process using the underlying logic. CIM provides two different kind of concepts to define aggregation and dependence associations, the CIM *Aggregation* and *Dependency* associations. Thus, the transitiveness of the privileges applied to an object can be incorporated by the rules depicts in figure 6.

1 $ManagedElement(?mea) \wedge AuthorizedTarget(?at) \wedge targetElement(?at, ?mea) \wedge privilege(?at, ?p)$
2 $ManagedElement(?meb) \wedge Aggregation(?ag) \wedge collection(?ag, ?mea) \wedge member(?mc, ?meb)$
3 $\rightarrow$
4 $AuthorizedTarget(?newat) \wedge privilege(?newat, ?p) \wedge privilegedElement(?newat, ?meb)$

1 $ManagedElement(?mea) \wedge AuthorizedTarget(?at) \wedge targetElement(?at, ?mea) \wedge privilege(?at, ?p)$
2 $ManagedElement(?meb) \wedge Dependency(?de) \wedge antecedent(?de, ?mea) \wedge dependent(?de, ?meb)$
3 $\rightarrow$
4 $AuthorizedTarget(?newat) \wedge privilege(?newat, ?p) \wedge privilegedElement(?newat, ?meb)$

Figure 6: *SWRL* rules for supporting hierarchical object access control

The two rules given in figure 6 are analogous, and the main difference between them is the association used in line 2. The first rule uses the aggregation association whereas the second rule uses the dependency association. Line 1 of both these rules retrieves any managed element ?*mea* which has a privilege associated. Line 2 retrieves any managed element ?*meb* which has an aggregation (first rule) or dependency (second rule) association with the previous managed element ?*mea*. As a result, the transitiveness infers a new privilege associated to the dependent or aggregated component ?*meb*.

Conditional RBAC [18] enables the assignment of roles and groups, and the granting of privileges, according to some conditions defined by the system administrator. This is the foundation of the so-called authorization policies [19], [20]. The expressiveness provided by the SWRL language has been used to provide a basic support for this kind of policies in this authorization model. This language allows the definition of conditional knowledge and for this reason, just by enabling the system administrator to insert her own SWRL rules in the authorization system, these rules can express the conditional nature of the authorization policies. This expressiveness in the authorization model enables it also to support constrained RBAC [21]. Constrained RBAC allows the expression of a condition that two incompatible roles cannot be assign to the same subject (separation of duties). This incompatibility can also be expressed in SWRL rules like any other policy according to the specific domain constraints. For example, a domain in which a separation of duties between *Role*(*DatabaseAdmin*) and *Role*(*WebAdmin*) has to be enforced can

use the SWRL rule depicted in figure 7 to control this separation. The rule retrieves all the identities belonging to the role *DatabaseAdmin* (lines 1-2) and all the identities belonging to the role $WebAdmin$ (lines 3-4) and if there is any instance belonging to both roles at same time (line 5) a new instance of the $Conflict$ concept will be inferred. The $Conflict$ concept has been defined semantically in OWL 2 to create an inconsistency in the inference process when any instance of this concept is created. Then, when the separation of duties constraint is violated an inconsistency arises and it causes the throwing of an exception which can be caught by the system administrator [22].

1 $Identity(?ia) \wedge Role(?ra) \wedge name(?ra, "DatabaseAdmin") \wedge$

2 $MemberOfCollection(?mca) \wedge member(?mca, ?ia) \wedge collection(?mca, ?ra)$

3 $Identity(?ib) \wedge Role(?rb) \wedge name(?rb, "WedAdmin") \wedge$

4 $MemberOfCollection(?mcb) \wedge member(?mcb, ?ib) \wedge collection(?mca, ?rb)$

5 $sameAs(?ia, ?ib)$

6 $\rightarrow$

7 $Conflict(?n) \wedge antecedent(?n, ?ra) \wedge consequent(?n, ?rb)$

Figure 7: Example of *SWRL* rule for defining a separation of duties between *DatabaseAdmin* and $WebAdmin$ roles

Finally, it is worth remarking the universality provided by the SWRL rules due to the semantics applied to the SWRL variables. These semantics produce the binding of the SWRL variables to *all* the instances available in the domain model; for example, the term $Identity(?x)$ is referred to *all* the identities available. Thus, if the domain is extended by other sources of information, the rule will change its scope and it will continue being applied to *all* the instances of the new domain.

As a result, a formal authorization model with support for RBAC, hierarchical RBAC, conditional RBAC, constrained RBAC, object hierarchy, subject hierarchy and separation of duties has been achieved by means of the usage of Semantic Web technologies.

5. Authorization Architecture for Distributed Systems

This section describes the architecture designed to apply the authorization model explained in section 4 on a distributed environment. A feature associated to the services of distributed systems is support for multi-tenancy [23]. This feature is defined as the ability to provide services to several *Parties*, simultaneously. In fact, figure 8 shows a distributed architecture with a multi-tenancy authorization system providing service to three different $Issuers$. An issuer is defined as the identity associated to a party and henceforth, both terms will be used indistinctly to refer to an independent information system which has its own *Users*, *Groups*, *Roles*, *Objects* and *Privileges*, that is, its own information model.

Regarding the different components available in the architecture, each issuer has two different components to

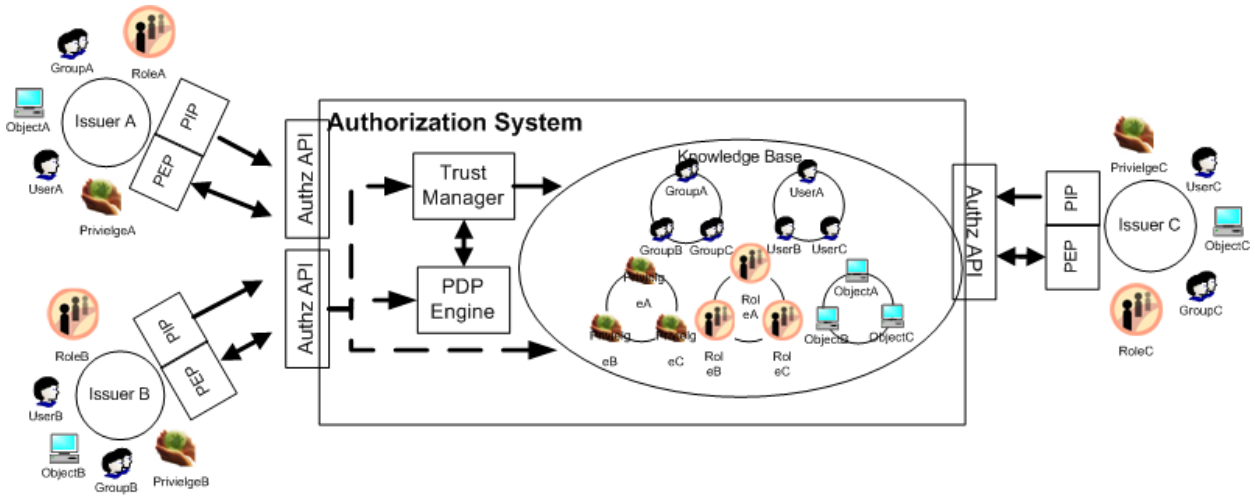

Figure 8: Authorization Architecture for Distributed Systems

interact with the authorization system: Policy Information Point (PIP) and Policy Enforcement Point (PEP). On the one hand, the Policy Information Point (PIP) component is in charge of providing all the issuer's information to the authorization system. This component provides an important added value in our system because it not only provides the current description of the information system (users, groups, privileges, resources) but also it is able to provide semantic information about this system. This fact enables the extension of the semantics of the resources being protected in the authorization system because the authorization system knows how to interpret this semantics and there is not a mismatch between the systems. Notice that this component provides all the information using the OWL 2 / SWRL languages. On the other hand, the Policy Enforcement Point (PEP) component is in charge of asking the authorization system about any authorization proof and it is also in charge of applying the result provided by the authorization system to the requesting issuer.

The authorization system is composed of two different modules. On the one hand, the Policy Decision Point (PDP) component where the authorization proof is created by applying the authorization model previously explained. This module receives an information model and it applies all the set of rules described in the section 4 to this model in order to determine the authorization statement. On the other hand, the authorization system provides a Trust Management (TM) component where all the trust relationships among the parties are managed. All the services of the authorization system are exposed to their users by means of an API which provides methods to use and manage the authorization system, the information used therein and the trust management.

Regarding security issues, the PIP-Authorization System and PEP-Authorization System communications may ensure the integrity, confidentiality and privacy of the data by means of the usage of digital signature and encryption methods such as XML-DS and XML-ENC, or by means of end-to-end encryption protocols like SSL and TLS. Moreover, all the issuers may be authenticated and the authorization system may control the access of the issuer to use the authorization system. This feature protects the authorization system against unauthorized issuers and it protects

against denial of service (DoS) attacks. Notice that there is a loose coupling between authentication and authorization systems and both $Users$ and $Issuers$ could be authenticated by any authentication mechanism, for example OpenID, X.509, SPKI/SDSI or Kerberos, among others.

5.1. Trust Management

The trust model of the proposed authorization architecture relies on the implicit trust of all the parties in the authorization system. Hence, the results provided by the authorization system, which in turn, might be authenticated for the parties, are trusted values. Moreover, the different issuers available in the distributed system can establish trust relationships among each other. In this context, a trust relationship, denoted as $A \preceq B$ (A trust B), represents a collaboration agreement for which issuer B is able to access to some knowledge of the information model of the issuer A. The negotiation protocols among the parties to establish the collaboration agreements, such as federation or coalition agreements, are carried out by external methods which will not be explained because they are out the scope of this paper. However, the authorization system might store individually the information models of each party in order to perform a dynamic management of the trust relationships and to be able to adapt the authorization system to changes in the trust relationships in run-time. In fact, by default no-one trust anyone else unless a trust proof is available in the authorization system, and only an issuer A can insert in or remove from the authorization system that $A \preceq someone$. All the stored information is depicted in the figure 8 as the knowledge base component. The trust manager is in charge of controlling the privacy of the entire information stored and selecting the information model to be used in the PDP component according to the trust relationship established.

The authorization system can offer two kinds of authorization processes. On the one hand, it can provide a regular authorization process to apply the authorization model by just taking into account the authorization information of the user who uses the system and ignoring the trust model. This case is used to provide a system that provides an authorization system that is totally independent for each party. On the other hand, a more interesting approach is to take into account the trust model. In this case, when an issuer asks about an authorization proof, the authorization system will compute the trust relationships of this issuer in order to generate the proper information model containing all the authorization information of those that trust this issuer. Then, this information will be used in the inference process of the authorization model. As a result, this process provides an authorization proof according to the information provided by the issuer and all those who trust this issuer, enabling a real collaboration among the issuers involved. Notice that authorization information is shared among issuers that establish trust relationships. Then, an issuer A could use the authorization information specified by another issuer B (assuming that B trust A) to carry out the authorization process. As a result, a distributed authorization model is defined among all the issuers that have established trust relationships.

5.2. Authorization Process

In order to explain the authorization process, let's suppose an authorization request in which an user belonging to an organization A in trying to access to a resource belonging to an organization B. Moreover, the organization A trusts

the organization B and vice versa ($A \precsim B$, $B \precsim A$). The sequence diagram for carrying out the authorization request is depicted in figure 9.

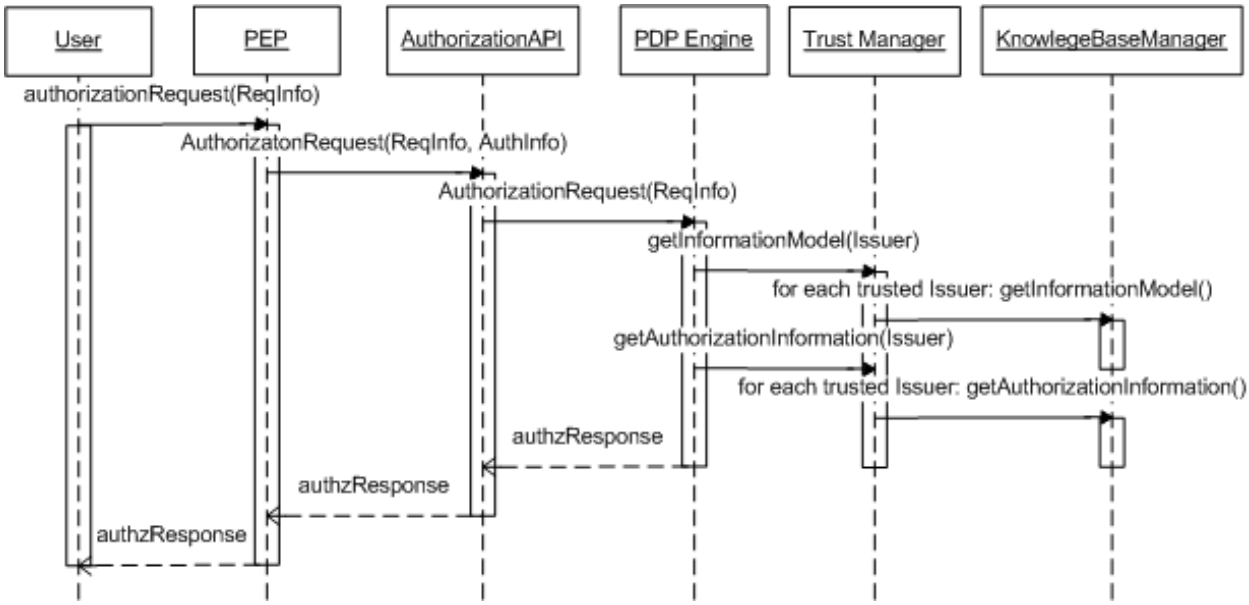

Figure 9: Sequence Diagram of an Authorization Request

Firstly, the user A tries to access to a protected resource controlled by the PEP of the organization B. Then, the PEP of the B organization establishes a secure communication with the authorization system using the *AuthorizationAPI*. This API exposes the authorization services to all the issuers. This API redirects the authorization request to the PDP component. The PDP component asks the *Trust Manager* (TM) for the information model to be used in the authorization process. The TM retrieves from the *Knowledge Base Manager* the information models involved in the authorization request . These models are: i) the information model of the organization associated to the protected resource; ii) the information model of all the organizations who trust this organization. In our example, the information of organization B and those who trust B are retrieved, which includes information of organization A. The *Knowledge Base Manager* is in charge of maintaining up-to-date the information model of all the parties in the architecture. To this end, either the *Knowledge Base Manager* periodically asks all the PIP components for updates in the information model (pull method) or the PIP component sends updates when there are changes in the system (push method). Once the *Trust Manager* has generated the proper information model according to the trust relationships established this information is sent to the PDP component. Then, the PDP provides an authorization statement applying the authorization model described in section 4. Finally, this response is sent to the PEP which is in charge of granting or denying the access to the resource according to this statement.

5.3. Privacy Management

To carry out the authorization process in a multi-tenancy environment will require the extension of the 3-tuple previously introduced in section 4 in order to take into account the different $Issuers$. The new 3-tuple can be represented as $(\{Issuer, Subject\}, \{Issuer, Privilege\}, \{Issuer, Object\})$ where $Subject$,$Privilege$ and $Object$ are now scoped by an $Issuer$ in order to identify the source of information of each element in the tuple. For example, the 3-tuple $(\{A, userA\}, \{B, privilegeB\}, \{C, objectC\})$ is interpreted as: B says that the subject $userA$ belonging to the party A has the privilege to carry out the action $privilegeB$ over the object $objectC$ belonging to the party C. Actually, this example entails that $A \precsim B$ and $C \precsim B$ because otherwise B will not be able to access the information model of A and C to establish the privilege using instances $userA$ and $objectC$, respectively. Moreover, the example also illustrates the privacy issues related to the visibility of the information to external issuers. The information shared with external issuers might be also negotiated in the collaboration agreement previously introduced in subsection 5.1. The previous example requires that the $userA$ and $objectC$ instances might be shared to create the 3-tuple. As a trivial solution, issuers can decide to share the whole information model related to the authorization information (in a glass box approach). However, in the practice, this model is not suitable in many scenarios in which the issuers try to keep to a minimum the information shared.

In order to minimize the information shared, the universality of SWRL rules can be exploited. Notice that a SWRL rule is applied to the overall information model. For example, the SWRL rule depicted in figure 10 produces that all the identities have the $access$ privilege to all the objects of the information model of the issuer $issuerA$.

1 $Identity(?i) \wedge ManagedElement(?o) \wedge issuer(?o, "issuerA")$

2 $\rightarrow$

3 $AuthorizedSubject(?as) \wedge privilegedElement(?as, ?i) \wedge privilege(?as, ?p) \wedge$

4 $Privilege(?p) \wedge name(?p, "access") \wedge$

5 $AuthorizedTarget(?at) \wedge targetElement(?at, ?o) \wedge privilege(?p)$

Figure 10: Example of rule for explaining the exploitation of the universality of SWRL rule

There is implicitly a natural extension of all the concepts unless some scope restrictions are done, such as the restriction of the $ManagedElement$ concept by means of the use of the $issuer$ property depicted in figure 10. This natural extension will be managed by the trust relationships established. Hence, let's suppose that the rule of the figure 10 is defined by the issuer $issuerA$. Let's also suppose that $A \precsim B$. Notice that while the $ManagedElement$ is restricted only to the $issuerA$ information system, the identities are not. Thus, when B asks for an authorization proof, the server will compute the information model according to the trust relationship and it will also take into account this rule provided by A. As a result, due to the universality of the rule and the non-restriction in the scope of the identities, all the identities belonging to both $issuerA$ and $issuerB$ will receive the privilege. Notice the suitability of this rule; it has produced effects on the identities of other parties, avoiding the necessity of sharing knowledge of the information

model related to these identities. This added value keeps to a minimum the information shared between the parties and helps to keep the privacy of the information.

6. Implementation

A prototype implementation of the CIM converter producing the OWL 2 representation of the information model described in the section 3 has been implemented as a proof of concept. In essence, it is a MOF-OWL converter based on a transform sheet using the extensible stylesheet language (XSL) to drive the process. The conversion process receives the MOF specification of the CIM schema and it produces an OWL 2 representation of this information model containing all the semantic features described in section 3.

The authorization system has been implemented as a prototype in order to validate both the authorization model described in section 4 and the authorization architecture suggested in section 5. This authorization system provides the authorization methods to be used for the different entities in the architecture by means of a REST [24] interface. Each PIP and PEP components provided by the different parties can use the authorization system by using the methods provided in the REST client. These methods enable the parties to access all the services of the authorization system remotely and securely.

The *Trust Manager* module has been implemented using the Jena [25] framework, which is a software able to manage knowledge bases of Semantic Web languages. Thus, the *Trust Manager* module stores individually the information of the all parties in the Knowledge Base. This information is provided by the PIP component. Then, the *Trust Manager* generates the proper information model to be used in the PDP component according to the trust information provided.

The PDP module which it is the core of the authorization system has been implemented using the Semantic Web reasoner Pellet [26]. This reasoner is able to deal with the semantic of RDF, RDFS, OWL 2 and SWRL languages. For this reason both the information model and semantics can been inserted therein to carry out the authorization and inference processes. This PDP component uses a configuration file which contains all the rules exposed in the section 4 to carry out this process. Moreover, Pellet also offers support to SPARQL queries used to generate the authorization proof.

As regard to the implementation of the PEP component, this is an API which each party has to use in order to protect their resources with our authorization system. In particular, this is a secure REST client to access to the methods provided by the authorization system.

The implementation of the PIP component of each party involved in the architecture relies on a CIM monitoring provider such as OpenPegasus or OpenWBEN together with a simple converter from the XML syntax they provide to the OWL representation of the current state of the system. Notice that the information provided by the CIM monitoring provider are only instances of the different concepts available in CIM schema and it does not provide additional semantic information like inheritance among concepts, inheritance among properties, restrictions, and etcetera. For

this reason, the PIP component also provides the semantic information associated to the resources to be protected in each the party, i.e. the information schema described in section 3. Indeed, it is important to remark that this information can be extended by the parties in order to insert semantic about the resources they are protecting. This enables to define any additional semantic information about the resources as well as to define any authorization policy associated to the conditional RBAC model. To this end, a SWRL/OWL 2 editor such as Protege [27] and *Ontology Rule Editor* (ORE) [28] can be used to define easily this semantics which it is inserted in the PIP component to be part of the information model. Notice the added value of providing methods to insert semantic information in the PIP component. At the end, each party is able to provide the current information model and the semantics in CIM-OWL 2/SWRL representation. All of this information is automatically processed by the Pellet reasoner inside of the PDP component. This provides as added value the capability to manage high level policies in the managed system whereas the semantic mismatch of the protected resources has been avoided.

As a result, the authorization system for distributed environment implemented is able to manage trust relationship in a multi-tenancy scenario and to use a formal authorization model with support for RBAC, hierarchical RBAC, conditional RBAC, constrained RBAC, object hierarchy, subject hierarchy and separation of duties. And it also is able to extend the semantic of the resources being protected and to manage high level authorization policies.

This authorization system has been successfully deployed in an intelligent building scenario. This is a pervasive environment in which there is monitoring devices in charge of the localization of users across the building as well as services provided by the building, such as projection services, room access, multimedia services, etcetera. Each of these devices runs an agent in a secure JADE [29] multi-agent platform which is in charge of the control of this device. Each agent uses a PIP component and a PEP component. Each agent represents a party of our authorization architecture. They provide the current status and the semantics of the resources to be protected in CIM-OWL 2/SWRL language by means of their PIP component. Additionally, there is an authorization agent in the multi-agent platform in which the authorization system is running. The authorization agent is in charge of providing an authorization service for the building services. Then, once a building user has requested to perform an action on a particular device, the PEP component that controls this device uses the REST API client to ask to the authorization agent about the access control statement. As a result, the PEP receives the answer from the authorization agent and enforces the resulting decisions, thus controlling the access to the resource. The answer is inferred using the authorization system described in this paper. Notice the added-value of the trust management in this scenario, since any device can specify that it trusts in other devices in order to share authorization information with them. As a consequence, all the resources available in the multi-agent platform of the building are protected by this authorization system.

7. Conclusions and Future Work

Semantic Web technologies have been identified as useful for authorization. A formal authorization model suitable for distributed system based on these technologies has been provided and validated by means of a prototype

implementation. The authorization model supports correctly some advanced features such as hierarchical RBAC, conditional RBAC, object hierarchies and separation of duties. A new conversion process of the CIM model to the Semantic Web languages has also been provided. As a result, a formal and highly expressive authorization system has been designed and validated as useful for distributed systems.

As future steps, it is expected to extend the authorization model with new features such as: the imperative aspects of the authorization policies; the capability to combine this authorization model with mandatory access control in order to create a formal hybrid authorization model; the ability to manage negative privileges; and the ability to incorporate PRBAC [18] features. Another excepted step is to classify and analyze the different syntactic and semantic conflicts that could arise in this authorization model and to provide formal methods for detecting and resolving them.

Acknowledgement

Thanks to the Funding Program for Research Groups of Excellence with code 04552/GERM/06 granted by the Fundacion Seneca. The authors would also like to thank the Spanish Ministerio de Educacion y Ciencia for sponsoring the research activities under the grant AP2006-4150 of the FPU program and for partial support of this research under the research project "Seiscientos. Infraestructura de Servicios Ubicuos y de Comunicaciones en Redes Vehiculares" with reference TIN2008-06441-C02-02. Finally, the authors would like to thank personally Nigel Edwards and Miranda Mowbray, researchers at Automated Infrastructure Labs, HP Labs for their valued comments on this research.

References

[1] S. Fugkeaw, P. Manpanpanich, S. Juntapremjitt, A Hybrid Multi-Application Authentication and Authorization Model using Multi-Agent System and PKI, Tech. rep., Proceeding at Communication Systems and Networks (2007).

[2] J. Li, D. Cordes, A Scalable Authorization Approach for the Globus Grid System, Future Generation Computer Systems 21 (2) (2005) 291–301.

[3] E. Palomar, J. M. Estevez-Tapiador, J. C. Hernandez-Castro, A. Ribagorda, Certificate-based Access Control in Pure P2P Networks, in: Sixth IEEE International Conference on Peer-to-Peer Computing, 2006.

[4] O. Canovas, A. F. Gomez-Skarmeta, G. Lopez, M. Sanchez, Deploying Authorisation Mechanisms for Federated Services in eduroam (DAMe), Internet Research 17 (2007) 479–494.

[5] A. Gbadegesin, R. Batoukov, D. R. R. , Flexible Scalable Application Authorization For Cloud Computing Environments, Tech. rep., United States Patent Application 20090228967 (2009).

[6] G. Klyne, J. J. Carroll, B. McBride, Resource Description Framework (RDF): Concepts and Abstract Syntax, Tech. rep., W3C (2004).

[7] D. Brickley, R. Guha, B. McBride, RDF Vocabulary Description Language 1.0: RDF Schema, Tech. rep., W3C (2004).

[8] B. Motik, P. F. Patel-Schneider, I. Horrocks, OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax, W3C Working Draft, W3C (April 2008).

[9] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission, W3C (May 2004).

[10] T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web, Scientific American May (2001) 10.

[11] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider, The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, 2003.

[12] W. Bumpus, J. W. Sweitzer, P. Thompson, A. R. Westerinen, R. C. Williams, Common Information Model: Implementing the Object Model for Enterprise Management, John Wiley & Sons, Inc, 2000.

[13] T. Vetterli, A. Vaduva, M. Staudt, Metadata Standards for Data Warehousing: Open Information Model vs. Common Warehouse Metadata, ACM SIGMOD Record 29 (3) (2000) 68 – 75.

[14] M. Majewska, B. Kryza, J. Kitowski, Translation of Common Information Model to Web Ontology Language, LNCS Computational Science - ICCS 2007 4487 (2007) 414–417.

[15] D. Heimbigner, DMTF - CIM to OWL: A Case Study in Ontology Conversion, in: Conference on Software Engineering and Knowledge Engineering, 2004.

[16] E. Prud'hommeaux, A. Seaborne, SPARQL Query Language for RDF, Tech. rep., W3C (2008).

[17] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, R. Chandramouli, Proposed NIST standard for Role-Based Access Control, ACM Transactions on Information and System Security 4 (3) (2001) 224–274.

[18] Q. Ni, E. Bertino, J. Lobo, S. B. Calo, Privacy aware Role Based Access Control, IEEE Security and Privacy 7 (4) (2009) 35–43.

[19] L. Kagal, T. Finin, A. Joshi, A Policy Language for a Pervasive Computing Environment, in: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks, 2003, p. 6374.

[20] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, J. Lott, KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction, and Enforcement, in: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks, 2003.

[21] R. Sandhu, D. Ferraiolo, R. Kuhn, The NIST Model for Role-Based Access Control: Towards A Unified Standard, in: Proceedings of the fifth ACM workshop on Role-based Access Control, 2000.

[22] E. Syukur, S. W. Loke, P. Stanski, Methods for Policy Conflict Detection and Resolution in Pervasive Computing Environments, in: 14th International World Wide Web Conference, 2005.

[23] C. J. Guo, W. Sun, Y. Huang, Z. H. Wang, B. Gao, A Framework for Native Multi-Tenancy Application Development and Management, in: 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services, 2007.

[24] L. Richardson, S. Ruby, RESTful Web Services, O'Really, 2007.

[25] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, K. Wilkinson, Jena: Implementing the Semantic Web Recommendations, in: Proceedings of the 13th international World Wide Web conference, ACM Press, 2004, pp. 74–83.

[26] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, in: Elsevier (Ed.), Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 5, 2007, pp. 51–53.

[27] N. Noy, M. Sintek, S. Decker, M. Crubezy, R. Fergerson, M. Musen, Creating Semantic Web contents with Protege-2000, IEEE Intelligent Systems 16 (2) (2001) 60– 71.

[28] A. Munoz, A. Vera, J. A. Botia, A. F. G. Skarmeta, Defining Basic Behaviours in Ambient Intelligence Environments by means of Rule-Based Programming with Visual Tools, in: J. C. Augusto (Ed.), 1st Workshp of Artificial Intelligence Techniques for Ambient Intelligence. ECAI, 2006.

[29] F. Bellifemine, G. Caire, A. Poggi, G. Rimassa, JADE: A Software Framework for Developing Multi-Agent Applications. Lessons learned, Information and Software Technology 50 (2008) 10–21.