

Context Ontology for Secure Interoperability

Céline Coma¹, Nora Cuppens-Boulahia¹, Frederic Cuppens¹, Ana Rosa Cavalli²

¹ GET/ENST Bretagne, 2 rue de la chataigneraie, 35512 Cesson Sevigne Cedex, France

² GET/INT Evry-CNRS, 9, rue Charles Fourier, 91011 Evry, France

Abstract

During interoperability exchanges, organizations are jointly conducting computation and sharing tasks. However, organizations can have different security policies. To guarantee good interoperability exchanges, organizations need to share with other participants information about the services they provide. In addition, to be compliant with security requirements during interoperability, security policies have to be dynamic. One purpose of this paper is to provide this dynamic behavior by taking care about context of access parameters. The context-aware security requirements may be met by using a contextual access control model to define the security policy of each party involved in the interaction, and OrBAC (Organization based Access Control) is an adequate model for this purpose. Elaborating an ontology based security model provides a mean to ensure sharing of understandable knowledge, in particular knowledge needed to derive the authorized accesses and usages during the interoperability sessions. In this paper, we thus suggest a context ontology to be combined with an ontological representation of the OrBAC model and show how it can be used to ease the security rules definition and derivation during interoperability sessions.

Keywords: *Security model, Interoperability, Ontology, Context, OrBAC*

1. Introduction

The growth of the Internet has triggered opportunities for cooperative computation and electronic interoperation (e-interoperation), where enterprises are jointly conducting computation and sharing business tasks based on the classified resources they each supply or authorize to use or to consult. These e-interoperations generally occur between organizations having different security policies. To conduct such interoperation, one organization must usually access the

resources of the other participants; however if no entity can be trusted enough to access or modify all the resources and whatever is the environment conditions (temporal, historical, spatial, etc.), context-aware access control is a primary concern. Moreover, support in the exchange of data, information, and knowledge is the key interoperation issue in current computer technology. Ontologies provide shared and common understanding of an application domain that can be communicated between organizations undertaking a collaboration. These ontologies play a major role in supporting secure e-interoperability. Most of current access control models do not support web semantics and ontologies to formally express security contexts and hence to ease the derivation of security rules in the case of interoperation. These ontologies are necessary to allow establishment of semantics compatibility between concepts used by organizations involved in a collaborative work. Semantics compatibility is needed to define or derive interoperation security policy used to accept or deny accesses.

We suggest in this paper a way to express collaborative contexts based on ontology definition and how they can be used to specify dynamic security policies. In Section 2, we analyze several contextual and ontology based access control models. Based on this analysis, we show that OrBAC [2, 14, 11] meets the requirements of flexibility and completeness we need to achieve our objectives. So we recall the main features of the OrBAC model in Section 3 and propose an ontological based description of OrBAC. In Section 4, we present our context taxonomy. Section 5 studies relationships between basic contexts. There are three types of relations: internal, composition and external. Definition of these relations and the preservation of ontology consistency are main challenges when creating our context ontology and defining security rules used to make access decisions. Section 6 presents ontological mapping which is used to define security policies of organizations that want to interoperate. In Section 7, we briefly show how to apply our approach in P2P collaboration. Fi-

nally, Section 8 concludes this paper.

2. Context and ontology in access control

Most access control models which take into account context are based on the R-BAC model [18]. Bertino *et al.* suggest extensions of R-BAC to take into account temporal constraints [3], spatial constraints [4] or both [5]. In TRBAC, the time is represented by periodicity, instant or temporal constraints. But in TRBAC, all permissions of a role must have the same time interval, this is a strong and not realistic constraint. Furthermore, this contextual access control model has problem with role hierarchy. It could have simultaneous occurrences of conflicting events and arbitrary triggering of interdependent triggers which can create ambiguity.

In case of geographical context, the most expressive location security model is GEO-RBAC. GEO-RBAC uses Geographic Markup Language (GML) [7] to express geographical constraints. Two types of geographical representations of spatial information exist in GEO-RBAC, absolute and logical representation. The first is a geometrical reference model (point for earth) and the second is a semantic location (a road, a university). This second representation has two parts: role schema and spatial role. The role schema describes the location perturbation technique to be applied to the instance of the role. A spatial role describes a user through a spatially bounded functional role.

GTRBAC is an extension of TRBAC that provides means to express geographical and temporal constraints. A time-based (or location-based) constraint is defined as a time (or location) event that causes a status event to occur. GTRBAC locations are structured in physical, logical and hybrid location hierarchies. This hybrid hierarchy and the separation of permission inheritance from role activation in GTRBAC can give rise to a complex semantics. Other extensions of GTRBAC exist. Chandran and Joshi stress on lack of flexibility and granularity of GTRBAC and have formalized an extension of GTRBAC, Lot-RBAC [6] for mobile service. Another approach by Bertino *et al.* is X-GTRBAC [5] which uses the concept of credential. Other models are based on trust as suggested by Wedde and Lischa [20]. But none of these access control models takes care of other contexts than temporal and spatial ones. That is why we consider the OrBAC model which has a more open view of context [11]. Another advantage of the OrBAC model is that contexts are associated with security rules instead of roles as in other aforementioned models. This provides means to specify more flexible security policies.

Taking into consideration context solves the prob-

lem of static policies but does not satisfy all collaboration requirements. Entities involved in a collaboration need common knowledge which should be understandable by all parties. An ontology provides an explicit conceptualization (*i.e.*, meta-information) that describes the semantics of the data and provides a common sharing knowledge which can be easily communicated. Some approaches like KAoS [19] and REI [16] provide semantic collaboration solution based on ontology. KAoS is designed to represent and reason on security policies of interoperable environments, such as Grid or Web services. For this purpose, KAoS is based on OWL [17]. But, KAoS is restricted to specification of security policies that do not require use of variable parameters. Security policies specified with REI are based on OWL-lite. OWL-lite is an ontological language which allows REI expression of security policies with constraints and obligations on environmental resources. REI extends OWL-lite language with logical variables to be more expressive. REI includes specification of meta-policy to solve conflicts. However, both REI and KAoS do not really take care about context and when they do, they consider a limited set of context types. For instance, REI ontology only deals with constraints and description of local entities.

3. Ontological representation of OrBAC

To deal with interoperability problems, we need an access control model which is dynamic and understandable by each party. In this paper, we choose the OrBAC model [2, 14]. The policy designer can use OrBAC to express security rules corresponding to permission, prohibition and obligation. The specificity of OrBAC is that traditional triples (subject,action,object) is abstracted at the organizational level into triples (role,activity,view). A role (respectively an activity and a view) is a set of subjects (respectively actions and objects) to which the same security rules apply. This specificity reduces the number of security rules to define and allows the policy designer to specify a security policy independently of its implementation.

OrBAC security policies are dynamic because OrBAC is a native contextual access control model. In OrBAC, each security rule is associated with a context [11] so that the security rule only applies when the context is active. For instance, in organization *P2PNet*, the role *peer* has the permission to do the activity *joinNetwork* on the view *networkS* in context of *shareFiles*. This permission is expressed by a rule: *securityRule(P2PNet,permission(peer,joinNetwork,networkS,shareFiles))*. Notice that the context *shareFiles* is not a spatial or temporal context

but a P2P application dependent context. This will be further explained in the following section. In OrBAC, a context type is modelled as a constraint that takes a subject, an action and an object as parameters. A context *Context* is active in a given organization when the predicate *hold* is true. $hold(?org, Context(?subject, ?action, ?object))$ means that within organization *?org*, context *Context* holds between the subject *?subject*, the action *?action* and the object *?object*.

Since a security policy in OrBAC can include security rules that correspond to permissions, prohibitions and obligations, conflicts between security rules can occur. In OrBAC, conflict management strategies can be defined to solve the possible conflicts. Such a strategy consists in assigning a priority to each security rule [9]. Priorities define a partial order on the set of security rules so that when a conflict occurs between two rules, preference is given to the rule with the higher priority.

To adapt OrBAC to interoperability objectives, we define an ontological representation of OrBAC using OWL DL [17]. We also redefine inference rules; for instance, to automatically derive a concrete permission that applies to a subject, an action and an object from an abstract security rule, we have the following rule:

$$is_permitted(?subject, ?action, ?object) \leftarrow \\ securityRule(?org, permission(?r, ?activ, ?view, ?cxt)) \wedge \\ hasProperty(?org, empower(?subject, ?r)) \wedge \\ hasProperty(?org, consider(?action, ?activ)) \wedge \\ hasProperty(?org, user(?object, ?view)) \wedge \\ hold(?org, ?cxt(?subject, ?action, ?object)).$$

In this inference rule, an organization *?org* specifies a security rule that corresponds to a permission for a given role *?r* to make a given activity *?activ* on a given *?view* in a given context *?cxt*. The properties *empower*, *consider* and *use* indicate that *?r*, *?activ* and *?view* are respectively abstractions of some *?subject*, *?action* and *?object* in the considered organization. When the considered context *?cxt* is being held for *?subject*, *?action* and *?object* through the *hold* predicate, we can thus derive the fact that it is permitted for *?subject* to make *?action* on *?object*.

In the following sections, we shall present the different contexts defined in OrBAC and suggest an ontological representation of these contexts compatible with the remainder of the OrBAC model.

4. Context taxonomy

An ontology is composed of two parts, a taxonomy and relationships between concepts of this taxonomy. So, we have first to create a taxonomy of contexts. We can see in Figure 1, a partial view of this taxonomy. A

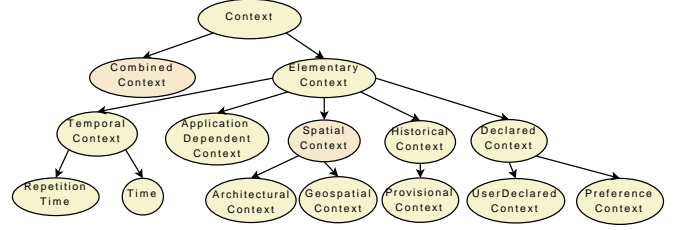


Figure 1. context ontology - context components

context may be a basic context component or a composed context. Composed contexts are presented in the following section. The main types of context component in our taxonomy are temporal contexts, spatial contexts, declared contexts, historical contexts and application dependent context.

The *temporal context* depends on the time at which a subject is requesting for an access to the system. Time ontology has already been defined in OWL-TIME [1]. OWL-Time provides the temporal concepts and relations for expressing temporal relations between instants, intervals, and events, together with information about durations, and about dates and times. It allows temporal predicates to apply directly to events. So, we use OWL-Time as an input of our ontology.

The Open Geospatial Consortium (OGC) has done research on geographic and spatial contexts. The ontological representation defined by OGC is very useful for geographical research, but is not fully adequate for security requirements. This is because, in our ontology, the *spatial context* may depend on the subject location but must also provide references to other contexts like network or sensor location. So, we need geographical information and the OGC provides an ontological input for this part but also architectural information. As architectural information, we list network type, address (IP, port), device type. Of course, we cannot use the same topological information for fixed wired and nomadic wireless networks. Since there is no standard ontology available to model architectural context, we have developed our own ontological representation for this part of our context taxonomy.

One context type, which is not considered by usual approach and is essential in security, is the *declared context*. The declared context depends on some declaration performed by subjects. We define different subtypes of the declared context. One of this subtype is the *user declared context*. In this case, the context depends on some characteristics related to the user's objective [11]. For instance, some subjects may be au-

thorized to declare that he or she is performing some access in the context (or purpose) of some collaborative activities, such as a conference, a course or a medical urgency. By declaring such a context, this subject will obtain specific permissions but also sometimes some obligations [10]. Of course, this subject must be first authorized to declare the context. For instance, only teachers may be authorized to declare the context of course and only physicians may be authorized to declare the context of medical urgency. Another subtype of the declared context is the *preferences context*. This context takes care about preferences such as versioning and language. Using this type of context, a subject can specify that he or she accepts interaction with another subject if the requirements associated with the preferences are satisfied.

The *historical context* is different from the temporal context. It depends on actions the subject performs in the information system. In our approach, we only consider historical context defined on previous actions also called *provisional context*. We do not consider historical context about future actions since we can always reformulate security rules that require future actions to be done into security rules that use provisional contexts [11]. When specifying security policy, the historical context is essential for usage control. Usage control advocates that to preserve security in information systems, we must check conditions which should be satisfied before the action is realized, but also conditions during and after the action.

Finally, the *application dependent context* depends on characteristics that join a subject, an action and an object. This context allows the policy designer to freely express security contexts related to the application. For instance, we could have medical, banking or administration applications. In medical application the context *consultation* requires that a physician is consulting the medical record of his or her patient; whereas in banking, the context *consultation* requires that a customer consults his or her account.

5. Ontological relationships and context compositions

After having structured our context taxonomy, we define relationships between contexts of this taxonomy to build our ontology. When creating the context ontology, we distinguish three relationship types, namely composition, internal and external. Composition and internal relationships are presented in this section whereas external relationships is explained in Section 6.

5.1. Composition relations of context ontology

The composition relations connect different contexts to define new composed contexts as shown in Figure 2. We consider two different types of composed context: logical composed context and ordered composed context.

Logical composition have already been studied in previous OrBAC context taxonomy [11]. In section 2, we claim that the combination of temporal and spatial context in GTRBAC raises problems. By contrast, in our ontology, the logical combination of the six types of aforementioned context is simply done by defining conjunction, disjunction or negation operators and using them to build logical composed contexts. Logical operators allows us to reduce the number of context creation and facilitates the expression of context combination.

Logical relations are not the unique links that must be integrated in the ontology. Historical contexts can be combined to obtain ordered composed contexts that are useful to define security policies in applications that require dynamic access control such as workflows. Thus, we can define privileges that only apply when some ordered composed context is active. For instance, in a P2P network, we can prohibit a peer to download files if the two following contexts occur in sequence: This peer first became a *goldMember* and then he or she downloads twice more files than it uploads. This is expressed in OrBAC as follows:
security_rule(p2pNet, prohibition(peer, download, files, isFollowedBy(goldMember, twiceDownloadthanUpload))).
 In this rule *isFollowedBy* is a sequence relation. *isFollowedBy(C₁, C₂)* means that context *C₁* is followed by context *C₂*. We can define other ordered composed contexts using relations such as *startWith*, *doneWith* or parallelism with *inParallelWith*.

As another example of composed context, we can also consider the relation *haveBeenDone(?ctx, ?nbTime)* which counts how many times a context has been activated (in this case the context is composed with itself). Using this composed context, a policy designer could give a permission only if this context has been activated a given number of times.

5.2. Internal properties of context ontology

The second kind of relationship is built on internal relations also called properties. When a location context overlaps another location context, overlapping is a property. For each context type, there

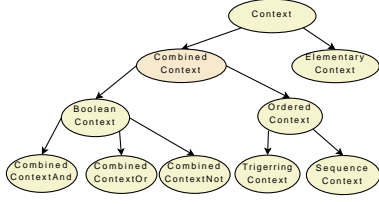


Figure 2. Context ontology – Context composition

are specific properties. In this section, we only give some internal properties to stress on compatibility relations. Our temporal context taxonomy is based on OWL-Time. So, we use OWL-Times properties such as *hasDurationDescription*. Because we want to detect and solve possible conflicts in our security policy, we use properties such as overlapping of time context to detect conflict, i.e. if the temporal contexts associated with two different security rules do not overlap, then there is no risk of conflict between these rules. Similarly, the spatial context is structured using internal relations between spatial contexts such as *outGeometricLocation*. Specific relations are also required for architectural context. For instance, we need internal relations to know if an information system contains a device *containsDevice*, or if a subject is in a network *memberOf(?subject, ?network)*. These relationships are generally useful to define application dependent contexts.

6. The ontology mapping

The external relation type is used to map different ontologies. It is partially derived from composition and internal relationships. Whereas current ontology mapping only stress on structure and semantic, our objective is to define security policies associated with interoperability exchanges. Thus, our mapping is also based on ontological relations from which we derive specific security relations as shown in section 6.4.

Let us consider two ontologies that respectively correspond to the security policies of organizations that plan to securely interoperate. Each ontology includes a context ontology as presented in the previous section. We divide ontology mapping into several parts. First part defines the terminological mapping T_{BOX} , which contains component classes, properties and hierarchical relations. The fact that *peer* is a subclass of *p2pNetwork* is specified in T_{BOX} . Second part defines the assertion mapping A_{BOX} which assigns instances to classes of T_{BOX} . The fact that *p1* is an instance

of class *peer* is specified in A_{BOX} . Third part defines the rule mapping R_{BOX} which provides rules to extend ontology expressivity. Fourth part creates compatibility relations between ontologies that take into account specificity of each organization and provides means to define interoperability security policies. In the following sections, we further explain these four mappings.

6.1. R_{BOX} mapping

To obtain the R_{BOX} mapping, we analyze taxonomies of the two organizations that need to interoperate and their *inference rules*. From internal and composition relations of the context ontology, we could derive new information on contexts in one or both of these organizations. For instance, if we consider a common knowledge about the existence of a hierarchical relation between two geographical contexts, the contexts *Barcelona* and *Spain*, this hierarchical relation is automatically derived by the following rule:

$$sub_context(?org, ?ctx1, ?ctx2) \leftarrow \\ inContext(?ctx1, ?org) \wedge inContext(?ctx2, ?org) \wedge \\ sameGeometricLocation(?ctx1, ?ctx2).$$

Let us assume that both organizations use the same context taxonomy. So, the inference rules associated with context ontology will be the same. In this case, we define a fact $contextRmatch(org, pred1(?args1), pred2(?args2))$ where $pred1$ and $pred2$ are context predicates, $?args1s$ and $?args2$ their arguments vertex, and $contextRmatch$ means that the two contextual predicates are compatible. If one of the two contextual predicates belongs to the other organization undertaking an interoperation, the same fact can be used to establish a compatibility and then derive the security rule to be activated. This compatibility will only be established after last step of our ontology mapping, which is the context revision.

6.2. A_{BOX} mapping

This second mapping is established by analyzing internal structures and correspondence between *attributes and their values*. A_{BOX} mapping is required for some global contexts such as geographical positions or used to compare entities and values contained in a declared context. This kind of mapping has already been studied in the context of ontology mapping, see for instance COMA++ [12] and GLUE [13], but without considering security issues. Notice that A_{BOX} mapping is easy to establish or to revoke when the key attributes are set up. Usually,

some similarity values belonging to the interval $[0,1]$, 0 for nearly dissimilar and 1 for nearly similar, are used to estimate the probability of correspondences. For instance, an organization should give a threshold for automatically deriving *contextAmatch* correspondences, in other words, semantically similar contexts. Obviously, context type is also taken into account in this mapping. *contextAmatch* can only occur when the two contexts have the same context type. We define two thresholds, *AmatchThreshold* for the matching of contexts and *AmismatchThreshold* for the mismatching of contexts. A_{BOX} mappings are established if the correspondence values belong to $[AmatchThreshold,1]$, thus the fact *contextAmatch(org, ctx1, ctx2)* is true. If these values belong to the interval $[AdismatchThreshold, AmatchThreshold]$, the mapping can be negotiated.

6.3. T_{BOX} matching

This is a *schema* mapping. We could, as we do previously, base T_{BOX} matching on existing works [12, 13, 15]. As T_{BOX} matching requires both schema and semantics correspondences, these works simplify and automatize search of context matching. An Or-BAC security policy is defined independently of its implementation. So, in order to preserve this suitable property in our ontology mapping, we have to define a more generic mapping than the classical mappings between two organizations. The reuse of previously global established mapping results simplify interoperability.

For instance, during file exchanges in a P2P network, several context mappings are similar. In this case, there are two ways to reuse these pre-established similar mappings: 1) we could define mappings common to all peers of the networks. These common mappings could be managed by a server, or 2) we could determine a mapping between two peers of the network and try to generalize this mapping to other peers. This allows us to define a hierarchy of mappings in the ontology structures.

[15] is a good approach to establish T_{BOX} matching. To establish when *contextTmatch(?org, ?ctx1, ?ctx2)* is true, A_{BOX} similarities have to be established first. In this case, three hierarchical mappings are possible: 1) the two mapped contexts have no sub_context. In this case, if *contextAmatch(?org, ?ctx1, ?ctx2)* is established then *contextTmatch(?org, ?ctx1, ?ctx2)* is established, 2) all sub_contexts of the two mapped contexts are globally mapped by A_{BOX} mappings, 3) the schema hierarchy of the two contexts are structurally similar and a set of sub_contexts are mapped by A_{BOX} mappings, even if immediate sub_contexts of the two

contexts are not.

6.4. Security compatibility relations

Let us consider two organizations that want to interoperate. We call the grantor organization the organization which grants access to its resources and the grantee organization the organization which gets access to the grantor resources¹. These two organizations have first to establish common knowledge as suggested in the previous sections in the case of mappings between contexts. That means mapping structure oriented concepts between these two organizations (contexts but also roles, activities and views) and then based on this common knowledge, each of them defines security compatibility relations between the grantor organization and the grantee organization. Thus, we have to map deontic oriented concepts between these two organizations (permissions, prohibitions and obligations). For this purpose, we use the approach suggested in the O2O model [?] as an interoperability framework. The O2O approach is based on creating a VPO (Virtual Private Organization). Each organization defines and manages its VPOs. A VPO *orgB2orgA* defines a security policy for subjects from *orgB* who want to have an access to some resources of *orgA*. This security policy is derived from local policy of the grantor organization. Actually, the VPO policy derivation process is based on compatibility relations. We have defined six compatibility relations: total, partial, symmetric, total open policy, partial open policy and context-aware. Due to space limitation, we shall one present the context-aware compatibility.

According to the previous mappings, context compatibility is established by rules as the followings ones:

$$\begin{aligned} & \text{context_compatibility}(?orga2orgb, ?ctxa, ?ctxb) \leftarrow \\ & \quad \text{contextTmatch}(?orga2orgb, ?ctxa, ?ctxb). \\ & \text{context_compatibility}(?orga2orgb, ?ctxa, ?ctxb) \leftarrow \\ & \quad \text{context_compatibility}(?orga2orgb, ?ctxac, ?ctxbc) \wedge \\ & \quad \text{contextRmatch}(?orga2orgb, \text{pred1}(?args1), \\ & \quad \quad \text{pred2}(?args2)) \wedge \\ & \quad \text{pred1}(?ctxac, ?ctxa) \wedge \text{pred2}(?ctxbc, ?ctxb). \end{aligned}$$

Once one of these rules matches, i.e. there is a mapping between a context from the grantor organization, *ctx-orga*, and a context from the grantee organization, *ctx-orgb*, and if a role compatibility relation is also established for some role from the grantor organization, *r-orga*, and some role from the grantee organization, *r-orgb* then if *r-orga* and *r-orgb* are context-aware compatible the security rule from the local security policy

¹Of course, in another interoperation, the grantor can become the grantee and the grantee can become the grantor.

of the grantor organization is added to the interoperability security policy of the VPO *orga2orgb* after revising the context *ctx-orgb*.

The operational definition of the context revision operator, \uparrow_c , depends on the context classification of the security rule (temporal, spatial, application dependent, historical or declared). For instance revision can be stated as follows:

$$\uparrow_c (in_secured_area) = in_confined_area$$

7. Contexts in P2P networks

P2P networks due to their complexity are good examples to show utility of context ontology in security policy. In P2P network, one common context to all kinds of interoperability is when some peer detects that one of the peers does not accept collaboration with other peers. This context will be simply expressed by an application dependent context *refusalCollaboration*. However, to reduce free-riding, this peer should get some penalty when this context is activated. If a peer disagrees to collaborate with more than half of the peers in a P2P system, this peer will have to pay or prove that it is a necessary node in the community otherwise it will be ejected from the network. Using OrBAC, this will be specified by obligations. These obligations can encourage a peer to limit the scope of its refusal to collaborate. So to avoid penalty, the suspected peer, *p1*, decides to collaborate with those peers belonging to the same network as *p1* and having the object it is looking for. In other cases *p1* prohibits access to its resources. So *p1* creates an application dependent context to know peers which could collaborate:

```
hold(?peer2p1, agreeCollaborate(?peer, ?action, ?obj))
← andContext(memberOf(p1, ?pnet),
  lookFor(p1, ?obj))∧
  context_compatibility(?peer2p1, ?peerNet, ?pnet)∧
  context_compatibility(?peer2p1, ?obj, ?searchByp1)∧
  andContext(memberOf(?peer, ?peerNet),
  source(?searchByp1, ?peer)).
```

which means that a peer *?peer* is a member of the *?peerNet* network which is compatible with *?pnet*, the network of *p1* and this peer *?peer* is a source for object *?searchByp1* which is compatible with the object *p1* is looking for. The context *lookFor(p1, ?object)* is an application dependent context which means that *p1* is looking for all objects *?object*. Obviously, we define the policy of peer *p1*, so that compatibility is established by the VPO defined by *p1*. After creating its context based on common knowledge of peers, the VPO of *p1* can define the following prohibition:

```
security_rule(?peer2p1, prohibition(peer_node, access,
```

```
resource, ↑c (notContext(agreeCollaborate(?peer))))).
This security rule means that in the VPO of p1, any
subjects assigned to peer_node are prohibited to per-
form any access on any resources in context of not
agreeing to collaborate.
```

8. Conclusion

Collaborative information systems must be both dynamic and secure. For this purpose, we need to define security policies that dynamically depend on events or time. The solution suggested in this paper is to allow the policy designer to express which security rules apply in various contexts. We first show that the OrBAC model is an appropriate model to achieve our goal and present an ontological representation of OrBAC. We then define an ontological based approach which is not restricted to temporal or spatial contexts but includes other context types. We describe our taxonomy of basic contexts and then analyze required relations to be able to express other types of composed contexts. Our context ontology model provides means to securely manage interoperability requirements. For this purpose, mapping between context ontologies has been defined. This mapping is based on detection of compatibility relations between ontologies and context revision operators. Context revision operators are used to adapt the mapping between security rules so that each organization involved in the interoperation can always enforce its security policy. This approach provides a framework to define interoperability security policies as suggested in the O2O model [8]. Collaborative activities in a P2P environment is used as an example to illustrate our approach.

Acknowledgment: For this work Céline Coma is funded by the Groupe des Ecoles des Telecom (GET) and Nora Cuppens-Boulahia and Frédéric Cuppens are partially funded by the DGE project P2Pim@ge.

References

- [1] Time Ontology in OWL. In J. R. Hobbs and F. Pan, editors, *Ontology Engineering Patterns Task Force of the Semantic Web Best Practices and Deployment Working Group, W3C notes*. w3c.org, September 2006.
- [2] A. Abou El Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, and G. Trouessin. Organization Based Access Control. In *4th IEEE Policy*, June 2003.
- [3] E. Bertino, P. A. Bonatti, and E. Ferrari. TRBAC: A temporal role-based access control model. *ACM TISSEC*, 4(3):191–233, 2001.

- [4] E. Bertino, B. Catania, M. L. Damiani, and P. Perlasca. Geo-rbac: a spatially aware rbac. In *10th ACM SACMAT*, June 1-3 2005.
- [5] R. Bhatti, A. Ghafoor, E. Bertino, and J. Joshi. X-GTRBAC: an XML-based policy specification framework and architecture for enterprise-wide access control. *ACM TISSEC*, 8(2):187–227, 2005.
- [6] S. M. Chandran and J. B. D. Joshi. *oT-RBAC*: A location and time-based rbac model. In *6th WISE*, November 20-22 2005.
- [7] O. Consortium. Geography Markup Language (GML) 2.0. In *OGC Document Number: 01-029*, February 20 2001.
- [8] F. Cuppens, N. Cuppens-Boulahia, and C. Coma. O2O: Virtual Private Organizations to Manage Security Policy Interoperability. In *2nd ICISS*, December 2006.
- [9] F. Cuppens, N. Cuppens-Boulahia, and M. B. Ghorbel. High-level conflict management strategies in advanced access control models. *ENTCS*, 186:3–26, July 2007.
- [10] F. Cuppens, N. Cuppens-Boulahia, and T. Sans. Nomad: A Security Model with Non Atomic Actions and Deadlines. In *18th IEEE CSFW*, pages 186–196, 2005.
- [11] F. Cuppens and A. Miège. Modelling Contexts in the Or-BAC Model. In *19th ACSAC*, 2003.
- [12] H.-H. Do and E. Rahm. COMA - A System for Flexible Combination of Schema Matching Approaches. In *28th VLDB*, August 2002.
- [13] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. Learning to match ontologies on the Semantic Web. *The VLDB Journal*, 12(4):303–319, 2003.
- [14] O. et al. *The OrBAC Model Web Site*. <http://www.orbac.org>, 2006.
- [15] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. In *27th VLDB*, pages 49–58, September 11-14 2001.
- [16] R. Masuoka, M. Chopra, Z. Song, Y. K. Labrou, L. Kagal, and T. Finin. Policy-based Access Control for Task Computing Using Rei. In *Policy Management for the Web Workshop*, pages 37–43, May 2005.
- [17] D. L. McGuinness and F. van Harmelen. Owl web ontology language overview. In <http://www.w3.org/TR/2004/REC-owl-features-20040210/>, February 2004.
- [18] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-Based Access Control Models. *Computer*, 29(2):38–47, 1996.
- [19] A. Uszok, J. M. Bradshaw, M. Johnson, R. Jeffers, A. Tate, J. Dalton, and S. Aitken. Chaos policy management for semantic web services. *IEEE Intelligent Systems*, 19(4), 2004.
- [20] H. F. Wedde and M. Lischka. Role-based access control in ambient and remote space. In *9th SACMAT Symposium*, June 2-4 2004.