

CONTEXT-AWARE ACCESS CONTROL USING SEMANTIC POLICIES

Anand Dersingh

Faculty of Computer Science,
Dalhousie University, Halifax,
NS, Canada
dersingh@cs.dal.ca

Ramiro Liscano

Faculty of Engineering and
Applied Science, University of
Ontario Institute of Technology,
Oshawa, ON, Canada
rliscano@ieee.org

Allan Jost

Faculty of Computer Science,
Dalhousie University, Halifax,
NS, Canada
jost@cs.dal.ca

ABSTRACT

One of the aspects of autonomic computing is self-protecting where systems are required to consistently enforce security policies in order to allow legitimate actions. The information that comes through the feedback loop after being monitored and analyzed tells the systems what is happening in the environments. The analyzed information describes situation and it is called context. The challenge lies in the question of how the autonomic system protects itself through changes of the situation or context, in other words, how access control policies can be properly written and enforced based on the context. Moreover, when the situation or context changes the policies must also reflect this change. A rudimentary approach is to manually write access control policies for all possible instantiations of the context. This is a cumbersome process and difficult to maintain with a large complex system. This paper focuses on access control policies and addresses these issues by representing context in semantic knowledge and extending a standard access control policy language, XACML, to incorporate the semantic knowledge. The work is validated by a proof of concept implementation.

Keywords: Context-Aware, Access Control, XACML, Ontology.

1 INTRODUCTION

In the world of computer communications, new technologies are developed and incorporated into existing systems at a rapid rate, resulting in increasingly complex, rapidly changing computer systems. Consequently, with such emphasis on new innovative features, long-term maintainability and sustainability are often ignored, resulting in ad-hoc systems where components, users, and services have been added and removed dynamically without any consideration of the system's overall integrity [2]. Thus, today's computer systems are often complex, have low cohesion, high coupling, and are becoming increasingly difficult to maintain [3].

The current approach to dealing with this increasing level of system complexity is to proportionally increase the number of administrative tasks, hence increasing the number of system administrators. However, this approach is unsustainable at best, as the growth of complexities will eventually make it impossible to match the number of system administrators with the complexity of computer systems [3].

Autonomic Computing is a potential solution to the problem of increasing system complexity and

costs of management. It is an approach where the ultimate goal is to create computer systems that can manage themselves while their complexities are invisible to end users [4]. This goal is similar to that of pervasive computing first expressed by Mark Weiser as: to create systems which are "so embedded, so fitting, so natural, that we use it without even thinking about it" [5]. Thus, from this common goal, by hiding and removing the low-level complexities from the realm of human control, humans can be freed from the burdens of management to concentrate on achieving higher-level, business oriented objectives.

In order to achieve this goal, it is necessary to have a system that is able to adapt its behavior relative to the context. This implies that the system must be able to understand and interpret policies that contain high-level meaningful terms [6]. This requires a high degree of semantic interoperability between the high-level policies and the changing context that drive such adaptation. A significant challenge is how to make a system with such a high-degree of semantic interoperability that it reduces the burden of system administrators to enable them to focus on high-level business objectives rather than low-level complexity management.

This paper addresses this challenge in the domain of context-aware access control by which an access control policy is created using high-level contextual terms and deployed on a policy system that controls accesses to resources or services. The contextual terms or contexts are natural human concepts about real world situations.

One of the contexts that is commonly known is the location context which has become popular due to the widespread availability of Global Positioning Systems (GPS). However, there are fundamental issues about this context. One of these is that location information as it is obtained from a sensor is relatively useless. Its context must be derived, i.e. a place, a street, etc. Thus, location by itself may not be suitable.

Another intuitive and interesting context is a user's activity. The user's activity increases the numbers of new context-aware applications. An example of such applications is sharing resources such as files or printers during an Instant Messaging (IM) session. It is possible to manage access rights to resources based on the IM session being in place. Access rights would then terminate when the context changes (the IM session ends.)

The object of this paper is to demonstrate how contexts can be captured and represented semantically, and integrated into an access control policy. In other words, how to make a policy system that understands and interprets high-level terms (contexts) used in an access control policy accurately in order to reduce burden of system administrators in managing the system.

This paper proposes a policy system that separates context management from access management. The policy system extends the eXtensible Access Control Markup Language (XACML) [7] by adding the capability of using context vocabularies in the policy and designating subjects and resources via semantic knowledge.

Related work is presented in Section 2. Section 3 introduces background material in Semantic Web technologies and policy-based network management including a brief introduction to XACML. The proposed approach is given in Section 4. Section 5 presents an example scenario. Sections 6 and 7 describe semantic context modeling and context-based access control, respectively, based on a given example scenario. Section 8 shows a proof of concept implementation. Conclusion and discussion of the paper is presented in Section 9.

2 PREVIOUS WORK

Chen et al. [8] concentrated on representing contexts in a formal way. The contexts under consideration were mainly basic concepts such as people, agents, places, and events. This work serves as a very first approach in using semantic

technologies for context representation. However, in many environments, it is obvious that these basic concepts alone are not sufficient. The system does not address the issues of context-based access control and how contexts can be integrated into a policy.

One research work in the area of context-based access control is Ubiquitous Context-based Security Middleware (UbiCOSM) [9] that adopts a context as a principle for security policy specification and enforcement process. Unlike the traditional access control model which is subject-based, access permissions are directly associated with contexts. UbiCOSM adopts an RDF-based format for context representation to cover heterogeneity of data representation. However, it does not extend the RDF-based format as a means to infer the relationships of entities. In other words, it does not support the deduction of higher-level context from the primary sensed context.

In terms of semantic policy languages, Kagal [10] proposed the Rei policy language which allows policies to be written using any semantic web language. Rei has been implemented in the N3 language and called the Rein policy framework [11]. The works presented in [12] and [13] are applications of Rei.

Another semantic policy language is KAoS [14] which is a framework for specification, management, conflict resolution and enforcement of policies. KAoS policies are based on OWL. KAoS uses Description Logic (DL) mechanisms to reason over policies in order to check for applicable policy as well as allowing for the classification of policy statements to enable conflicts to be discovered.

None of the mentioned policy languages address the clear separation between context modeling and access management, resulting in poor performance because the context has to be evaluated at run-time as well as confusion differentiating between access policies and context deriving rules.

The approach taken in this paper allows an administrator to focus on either access management or context modeling separately. This is important because an administrator does not have to worry about the meaning of the contexts when writing an access control policy. Besides, the meaning of the contexts can be changed or added without changing the policy.

This paper is similar to the work done by E. Damiani et. al. [15], since they also extend XACML to make it semantic-aware. However, the approach taken in this paper is different. First, this paper uses OWL as a representation language of the subject and resource as opposed to using only RDF. Second, this work deals directly with the contextual information that is associated with a subject and resource. In addition, this paper separates context management from access management. The extension of XACML in this paper is done by leveraging attributes of the

XACML subject and resource specification in order to carry associated contexts.

T. Priebe et. al. [16] have proposed an extension to XACML using attributes as well. However, their approach adds semantic inference at the point of the access control decision. The approach taken by this work is to perform semantic inference prior to access time, thus significantly reducing the time it takes to process an access request. In addition, it is worth noting the work from A. Toninelli et. al. [17] which is similar to our work in terms of context modeling, but again in terms of policy evaluation it still adds the cost of inference during access time.

3 BACKGROUND

3.1 Knowledge Representation: A Semantic Web Approach

Access control is a mechanism of determining whether a request to resources and data in a system should be permitted or denied. Several access control models have been developed based on different schemes. These models include Discretionary Access Control (DAC), Mandatory Access Control (MAC) and Role-based Access Control (RBAC) [18].

In autonomic computing and context-aware systems, the mentioned access control models may not be appropriate because the systems should be protected based on situational contexts (or feedbacks from the control loop [1]). Context-aware access control systems have been proposed such as [16], [17], [19], [25] which are more appropriate than using the traditional models. However, contexts can be anything such as location, time, activity, identity etc., and it is important that an access control system understands and interprets these contexts accurately.

Ontologies are the key to representing vocabularies in a formal way for the following reasons [8]: 1) a common ontology enables knowledge sharing in an open and dynamic distributed system, 2) semantically well-defined ontologies provide a means to reason about contextual information, and 3) explicitly represented ontologies allow interoperability among devices.

Although XML DTDs and XML Schemas are sufficient for exchanging data between parties who have agreed to definitions beforehand, their lack of semantics prevents machines from reliably performing this task when presented with new XML vocabularies. The same term may be used with (sometimes subtle) different meaning in different contexts, and different terms may be used for items that have the same meaning. RDF and RDF Schema approach this problem by allowing simple semantics to be associated with identifiers. With RDF Schema, one can define classes (concepts) that may have multiple subclasses and super classes, and can define properties, which may have sub properties, domains,

and ranges. In this sense, RDF Schema is a simple ontology language. However, in order to achieve interoperation between numerous, autonomously developed and managed schemas, richer semantics are needed. For example, RDF Schema cannot specify that the Person and Car classes are disjoint, or that a string quartet has exactly four musicians as members [20].

OWL[26] has more facilities for expressing meaning and semantics than XML, RDF, and RDF-S, and thus OWL goes beyond these languages in its ability to represent machine interpretable content. OWL is a revision of the DAML+OIL web ontology language incorporating lessons learned from the design and application of DAML+OIL [21]. OWL specifies descriptions for the following kinds of concepts [21]:

- Classes (general things) in the domains of interest
- The relationships that can exist among things
- The properties (or attributes) those things may have

Ontologies are usually expressed in a logic-based language, so that detailed, accurate, consistent, sound, and meaningful distinctions can be made among the classes, properties, and relations. Some ontology tools can perform automated reasoning using the ontologies, and thus provide advanced services to intelligent applications such as conceptual/semantic search and retrieval, software agents, decision support, speech and natural language understanding, knowledge management, intelligent databases, and electronic commerce [21].

3.2 Policy-based Network Management

Policy-based Network Management (PBNM) has been proposed as a solution to network management problems such as network congestion, administrative challenges, network complexity, security, etc. PBNM is considered as a paradigm in network management by managing the network as an entity itself instead of managing the individual components. This similar policy-based approach has also been used in autonomic computing systems.

Policies are commonly used in autonomic computing systems to automate system behavior while reducing or perhaps eliminating human intervention. Policies are business objectives that are expressed in terms of logic and actions or behavioral rules to be performed in response to certain situations which is decoupled from implementation mechanisms. The mechanisms are usually fixed at design time but the actual run-time behavior must adhere to the policies.

The two main architectural components of the PBNM are the Policy Enforcement Point (PEP) and Policy Decision Point (PDP). Figure 1 shows a simple configuration involving these two

components [22].

The PDP is a process that makes decisions based on policy rules and the state of the services those policies manage. In most cases, the PDP transforms the policy decisions and passes them to the PEP in a form and syntax that the PEP can accept. However, the PDP may use additional mechanisms and protocols to perform additional tasks such as retrieving policy information i.e. from a policy repository.

The PEP represents the component that always runs on a policy-aware node e.g. device. It is the point that enforces a policy decision and makes configuration changes according to the policy decision. Network nodes can be categorized into groups: policy-aware and policy-unaware. For policy-aware nodes, the PEP is usually a component residing in the nodes. But policy-unaware nodes are nodes that are unable to interpret or form policy information. However, they can still participate in a policy-based network if there is an enforcement mechanism that performs translation from policies to device configurations. As depicted in Figure 1, the translation is performed by a policy-aware agent.

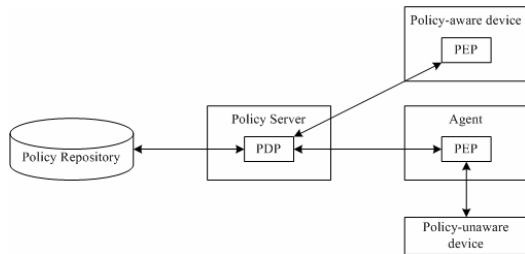


Figure 1: General architecture of PBNM [22]

According to the IETF Policy Based Network Management MIB [23], the interaction between the PDP and PEP starts with the PEP that receives a notification or a message or an event that requires a policy decision. The PEP then formulates a request based on the notification received and sends it to the PDP. The PDP consults the policy repository, makes decision, and returns the policy decision to the PEP. The PDP may also return additional information that may or may not be associated with the decision.

The language used to specify the policies implemented by the PDP is also of vital importance. The number of policies applicable at a network node might potentially be quite large. At the same time, these policies will exhibit high complexity, in terms of the number of fields used to arrive at a decision. Various policy languages have been proposed such as Ponder [24], XACML [7], Rei [10], KAoS [14], etc. These policy languages ensure a coherent and consistent application of policies as well as ensure unambiguous mapping of a request to a policy action. They also permit the specification of the sequence in which different policy rules should be

applied and the priority associated with each one.

The policy language used in this work is XACML which is a policy language in the form of XML for expressing access control policies. It describes policies and rules in terms of Boolean combinations of attribute-value pairs based on the subject, resource, and environment. It also provides conflict resolution through its combining algorithms.

The XACML language is defined by two XML schemas: “*xacml context*” and “*xacml policy*”. Fig. 2 depicts XACML context. The “*xacml context*” defines how to represent policy request and policy response messages exchanged between the PEP and the PDP. The “*xacml policy*” defines how to represent the access control policies.

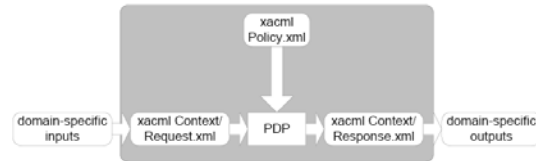


Figure 2: XACML Context [7]

4 APPROACH

The complexity of autonomic computing can be managed using a policy-based approach of which context-based access control is a part. Thus the analyzed information or contexts can be used directly in access control policies. In order to create a policy that conforms to business-level policies or agreements, contexts should be derived from high-level concepts. This paper addresses this issue by proposing a way in which a context-based access control policy should be created as shown in Fig. 3.

At first it is necessary to have an agreement on high-level vocabularies (situational contexts) as shown at the top of Fig. 3. The agreement should be made from the perspective of the business, security and users to define contexts of the situation in order to govern the system. Then it is necessary to have a semantic context layer, in other words, the meaning of the contexts. This is crucial since it affects the entire access control system if one does not have a clear definition of the contexts.

Once the meanings of the contexts have been clarified, it is necessary to create a context ontology. The ontology defines common words and concepts (meanings) used to describe and represent an area of knowledge. It also includes machine processable definitions of basic concepts and the relationships among them. The terms (contexts) defined in the ontology will be used to construct a context-based policy.

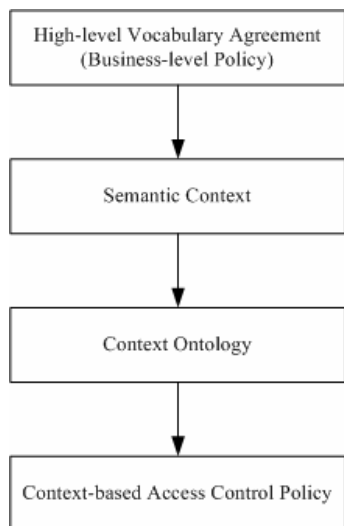


Figure 3: Process of creating a context-based access control policy

4.1 Context Management

This work uses the power of Semantic Web languages in order to achieve goals in designing vocabularies for a context-based access control policy. The goals are 1) representing vocabularies in a formal way so that they are processable by a policy system, 2) the vocabularies can be easily shared and reused perhaps for other systems or applications and 3) a new vocabulary can be added at later time without disruption to a system that is deployed and running.

4.1.1 Vocabulary Representation

There are two types of vocabularies that can be designed in an ontology. The first type of vocabulary is the name of the classes (concepts) themselves. Classes may be authors, books, publishers, places, people, hotels, and so on. Basically, it is an object or thing that one wants to talk about. For example, one can define a class Person which keeps persons' identities such as name, email, ID, etc. One can also have subclass-superclass relationships for each class. A class Person may have subclasses which may be categorized by roles in an organization such as in a university where roles could be "Student", "Professor", "Staff" etc. One can also classify by sex, marital status, or even biological information. This is entirely up to the domain of interest.

Another type of vocabulary is a property name that relates two classes together. Again one can have any name that relates two things (classes) together, for example age, title, in a room, in a meeting, and so on. Both types of vocabularies can be represented using OWL.

4.1.2 Context Assertion

Representing a context ontology using OWL provides the structure of how contexts relate to each other. However, a context itself cannot be used without its assertion. An assertion is the process of acquiring and categorizing domain knowledge into a semantic knowledge base. As mentioned, there are two types of vocabularies: a class itself, and a relationship. In order to assert an instance of a class, one can actually integrate the knowledge base to external entities such as RFID, LDAP, etc. Reasoning over classes is also a crucial part.

One can use the reasoning power of OWL to reason over defined vocabularies. One can reason about:

- Class membership: If x is an instance of a class C , and C is a subclass of D , then one can infer that x is an instance of D
- Equivalence of Classes: If Class A is equivalent to class B , and class B is equivalent to Class C , then A is equivalent to C as well.
- Consistency. Suppose one has declared x to be an instance of the class A and that A is a subclass of $B \cap C$, A is a subclass of D , and B and D are disjoint. Then one has an inconsistency because A should be empty, but has an instance x . This is an indication of an error in the ontology.
- Classification: If one has declared that certain property-value pairs are a sufficient condition for membership in a class A , then if an individual x satisfies such conditions, one can conclude that x must be an instance of A .

However, there is a limitation in OWL reasoning. The language of OWL (Lite and DL) can be viewed as specializations of predicate logic which is illustrated by logic axioms. Another subset of predicate logic is a rule system (Horn Logic). A rule has the form: $A_1, A_2, \dots, A_n \rightarrow B$ where A_i and B are atomic formulas. Description logic (OWL) and Horn logic (rules) are orthogonal in the sense that neither of them is a subset of the other. For example it is impossible to assert that persons who study and live in the same city are "home students" in OWL, whereas this can be done easily using rules: $\text{studies}(x,y), \text{lives}(x,z), \text{location}(y,u), \text{location}(z,u) \rightarrow \text{homeStudent}(x)$ [20]. On the other hand, rules cannot assert the information that a person is either a man or a woman, whereas this information is easily expressed in OWL using a disjoint union.

4.1.3 Context Management Framework

Fig. 4 shows a context management framework. A Context Manager is a mediation mechanism between real world raw information and a semantic knowledge base. It is composed of two processes: a context acquisition and a rule manager. Context Acquisition is a process that gathers information

from the outside world and performs instance assertions into the semantic knowledge base. These assertions are those that do not require inference capability but can be asserted based on context categorization. This paper uses the term “direct assertion” for this type of assertion. After performing direct assertion, this triggers the Rule Manager in order to infer over instances that have been asserted by the context acquisition. The results of inferring direct asserted instances are asserted into the semantic knowledge. This paper uses the term “indirect assertion” for this type of assertion.

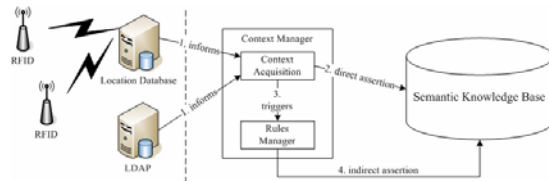


Figure 4: Context management framework

4.2 Context-based Access Control Policy

Policies are usually written in the form of rules. Each policy must be associated with domain knowledge. In context-aware access control, a policy must be able to use the contexts of the situation in order to perform access control. Therefore, it is necessary to reason over the domain knowledge to obtain the contexts. Fig. 5 depicts the relationships among an access control policy, policy rules, and a context ontology.

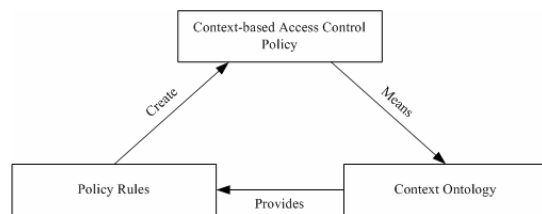


Figure 5: Construction of a context-based access control policy

The policy may or may not require situational contexts or defined contexts to allow access. If the policy requires defined contexts, it will be written using context vocabularies directly into the policy. For example, a policy for users engaging in a particular activity would be “if any persons are in a consultation, then they are allowed to access resources located in the consultation room.” Here the vocabulary is “in a consultation” which has to be semantically defined in the ontology.

The policy must be written with conformance to the contexts defined in the previous section. In order to achieve that, the policy language used in this case must be able to understand and interpret semantic representation of the vocabularies.

One approach in order to easily integrate contexts into an access control policy is to insert rules that infer each context into an access control policy. This is a cumbersome process and error prone since every time a context is needed in a policy it requires inserting these rules into the policy. This paper addresses this issue by using a semantic knowledge base that maintains information about the contexts (defined vocabularies) and lets an access control policy use terms defined and asserted in the semantic knowledge base directly without writing rules to interpret contexts again.

5 EXAMPLE SCENARIO

In context-aware access control, what is essential is the context of the situation. An example is an Intensive Care Unit (ICU) patient-doctor scenario where a patient is under the care of an ICU doctor and their vital signs are being monitored. Assume that, during the visit, the ICU doctor needs to consult an expert due to some abnormality detected in the captured patient’s physiological parameters such as temperature, heartbeat, blood oxygen, blood pressure, etc. This scenario mimics a “real” of premature babies and ill term babies in rural and remote hospitals that lack of Neonatal Intensive Care Units (NICUs) [33] that scenario the attending physician must consult a neonatologist via a telephone call, who may or may not be located at the NICU at that time, and has to describe the baby’s condition verbally. The consulting neonatologist receives physiological data (from sensory devices attached to the baby) through his PDA, laptop, or PC and uses this data to advise the physician [33].

A significant challenge in this scenario is to control who can see the data and under what conditions. Due to the ability of remotely accessing the patients vital signs it is important that this information not be remotely accessible at all times the patient in the ICU, but perhaps only under particular situations like a consultation. In other words, access to the patient’s real-time data can be controlled based on the situational context. It is also important to note that this scenario is ad hoc because it is impossible to predict who the doctor will consult and if and when the consultation will occur. Therefore it is not possible to use conventional subject or role-based access control approaches in this case. In the given scenario, resources will be shared based on implicit human knowledge such as the same location, in a phone call, or in a consultation. This implicit knowledge needs to be captured and represented by the system in a formal way as contexts in order to readily share resources.

Even though this example is not a typical autonomous computing example it shares the common conceptual control loop feature of autonomous computing [1] in that the implicit human knowledge

(and domain knowledge) needs to be monitored and analyzed, and an appropriate action must be taken based on a set of policies (planning process). It is also important to note that there is an acceptance in the autonomous computing community that contextual-based access control is important but there appear to be little or no examples presented of contextual-based access control in autonomous computing and examples that are presented typically are from the pervasive and ubiquitous computing domain.

A contextual consultative scenario is shown in Fig. 6. In this scenario, a patient, Jane, is in an ICU room at a medical center with an ICU doctor, Alice. There is an ICU monitor attached to Jane in order for Alice to monitor physiological data. This ICU monitor is deployed as a Web Service in order to facilitate remote access to it. Alice requires further assistance from an expert, Bob, by consulting Bob via a call (note this call could be a multimedia call made through the ICU monitoring device or a telephone call). Alice verbally tells Bob to access the web service via his browser to receive and analyze real-time physiological data and further advises Alice. There are two participants physically in the room, Jane and Alice. The room is equipped with RFID location sensors that can detect persons in a room. Each person also has an RFID tag. In addition, all calls made from the room are monitored. Bob, participates in the consultation via a teleconference call from a remote site.

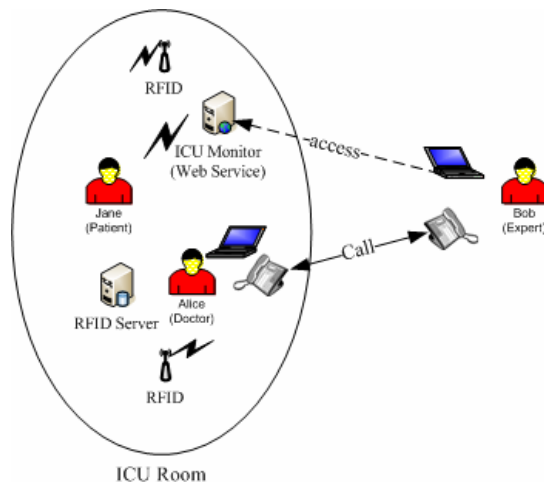


Figure 6: Contextual consultative scenario

The goal is to permit the sharing of resources among the participants, whether or not they are at the same site, in a consultative session such that they conform to acceptable enterprise policies. An example of this type of policy is “*Any persons participating in a consultation are allowed access to resources located in the same consultation*”. For this particular scenario there may be other policies in

place that may dictate who can create a consultation, for example only an ICU doctor may do this, and this can be easily done with a conventional role-based access policy and is not the thrust of this paper.

This is a high-level policy as compared to typical network level policies that govern a particular person’s access to some resource. The immediate questions that arise from such a policy are the meaning of the terms “in a consultation” and “in the same consultation”. These terms are the contexts that need to be defined and we have chosen a semantic representation of these terms. In addition, how these high-level terms can be used directly in an access control policy and recognized by a policy system for evaluation.

6 SEMANTIC CONTEXT MODELING

For demonstration purposes, this paper uses the policy presented in the previous section and is applicable to the collaborative scenario shown in Fig. 6. Below is a policy pseudo code of such a policy translated from the actual XACML policy (shown later in Fig. 11.) Note that this pseudo policy form is for the purpose of readability, and is not an actual executable XACML policy.

```

Rule 1
If {
  Subject == AnyPerson
  Resource == AnyResource
  Action == AnyAction
Condition {
  AttributeValue(SubjectAttribute(inConsultation)) ==
  AttributeValue(ResourceAttribute(inConsultation))
}
}
Then Permit

```

In this policy, the term “inConsultation”, needs to be defined. In the process of doing this other significant intermediary contextual terms particular to the situation also get defined. These are “inCall”, “inSameCall”, and “hasLocation”. Here “hasLocation” is a primary context. A primary context is a context that can be captured directly from a sensing device. “inCall” and “inSameCall” are secondary contexts derived from primary contexts (Rules 1 and 2 below). “inConsultation” is also a secondary context derived from primary and secondary contexts (Rules 3 and 4 below). Contexts here are vocabularies; however, for the purpose of the paper a context refers to a vocabulary of a real world situation that will be defined in an ontology and used in an access control policy. Other vocabularies in the ontology refer to conceptual objects in the world. The process of defining entities in an ontology is called ontology engineering.

As mentioned in section 4, there are two representations of a vocabulary or context in an

ontology: these are by classification (classes) and by relations between classes (properties). Fig. 7 shows entities defined as classes and their subclass relationships in the context ontology.

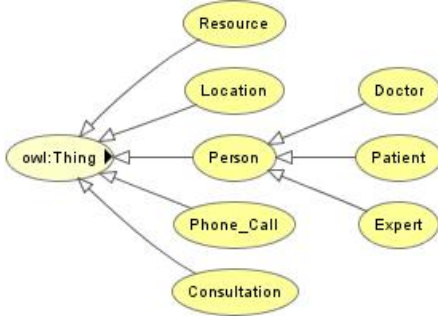


Figure 7: Classes and subclasses relationships in a context ontology

A consultation, a person, a resource, a phone call and a location are defined as classes. Person class has three subclasses: doctor, patient, and expert. In the scenario, an actual phone call is based on the establishment of a call session using the Session Initiation Protocol (SIP) since it allows different types of media (i.e. voice and video) to be part of a session. The following properties are also defined: “inCall”, “inSameCall”, “inConsultation”, “hasLocation” as shown in Fig. 8 as dashed lines. These properties are actually the contexts of the situation that need to be captured and used in a policy. “inCall” means a person in a call (teleconference). “inSameCall” means two or more persons are in the same call. “inConsultation” means an entity is in a consultation. Here the entity could be a person, agent, or resource. “hasLocation” identifies the location of an entity. Fig. 8 shows some classes and properties in the context ontology. The “isa” relationships mean “is a subclass of”. Each dashed line is a property that relates two classes together.

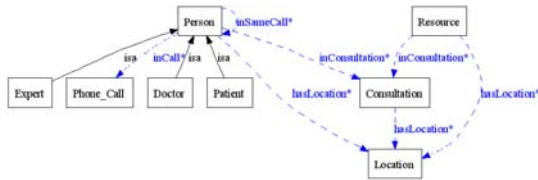


Figure 8: Classes and properties related to the consultation context

It is necessary now to assert instances (facts) based on the relationships defined in Fig. 8. There are two types of assertions. For the location, one can use location-sensing devices in an ICU room and assert information obtained from the devices as an instance of a class “Location”. Therefore, the assertion of the context “hasLocation” can be

asserted based on the sensing devices. This is a direct assertion of knowledge. Note that in this scenario the access control system is at site B and cannot capture location from site A which means one cannot capture Bob’s location in this case.

For the phone call, since a SIP-based approach is used, the Call-ID of the session obtained from the SIP Proxy can be used as a fact and asserted as an instance of type string in the “Phone_Call” class.

Another type of assertion is indirect assertion by using a rule system. This type of assertion instantiates instances of properties like context “inConsultation”. This work uses Semantic Web Rule Language (SWRL) [27] to capture the meaning of relations between classes. Rule 1 captures a person in a call. Rule 2 captures the meaning of persons in the same call by using Call-ID of the teleconference. Rules 3 and 4 capture the meaning of persons in a consultation. The term “in a consultation” means physically located in the consultation room or connected via a SIP call which are captured by rules 3 and 4 respectively. Rule 5 captures the meaning of resources in a consultation based on location.

Rule 1: capturing a person in a call

$$Person(?person_x) \wedge Phone_Call(?call) \wedge hasCallID(?person_x, ?ID) \wedge hasCallID(?call, ?ID) \rightarrow inCall(?person_x, ?call)$$

Rule 2: capturing persons in the same call

$$Person(?person_x) \wedge Person(?person_y) \wedge Phone_Call(?call) \wedge hasCallID(?call, ?ID) \wedge hasCallID(?person_x, ?ID) \wedge hasCallID(?person_y, ?ID) \rightarrow inSameCall(?person_x, ?person_y)$$

Rule 3: capturing a person who is participating in a consultation based on location

$$Person(?person_x) \wedge Location(?room) \wedge Consultation(?con) \wedge hasLocation(?con, ?room) \wedge hasLocation(?person_x, ?room) \rightarrow inConsultation(?person_x, ?con)$$

Rule 4: capturing a person who is participating in a consultation based on a SIP call

$$Person(?person_x) \wedge Person(?person_y) \wedge Location(?room) \wedge Consultation(?con) \wedge hasLocation(?person_x, ?room) \wedge hasLocation(?con, ?room) \wedge inSameCall(?person_x, ?person_y) \rightarrow inConsultation(?person_y, ?con)$$

Rule 5: capturing a resource located in a consultation room (ICU room)

$$Resource(?resource) \wedge Location(?room) \wedge Consultation(?con) \wedge hasLocation(?conf, ?room) \wedge hasLocation(?resource, ?room) \rightarrow inConsultation(?resource, ?con)$$

Note that resource capturing can be done

automatically by using rule 5 or it can be done manually. Since it might be the case that not every resource will be part of the consultation.

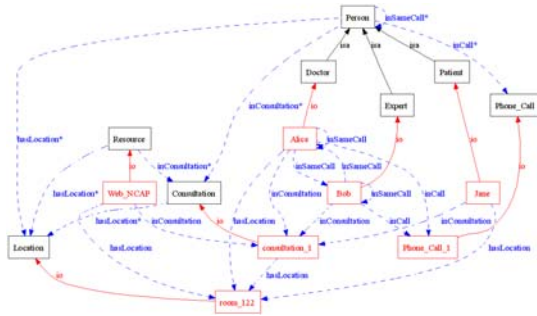


Figure 9: Instances assertion as a result of SWRL rules with their classes

Fig. 9 and Table I show assertions of instances with their classes based on the above rules in graphical representation and in a persistent database form (partially displayed) respectively. In this example we use a Web NCAP [32] as an example of an ICU monitoring Web Service. The Web NCAP is a Web Service implementation of the IEEE 1451 compliant Network Capable Application Processor (NCAP) instrumentation gateway. Our Web NCAP is interfaced to a wireless sensor network that can monitor temperature and motion.

The way that the knowledge is stored in a relational database is in the form of triples i.e. subject-property-object. In Fig. 9 an “io” relationship means an instance of a class. “isa” means “is a subclass of” and each dashed line is a defined property. One can see that there is a consultation named “consultation_1”. After inferring the above rules one can see that:

- Rule 1 asserts Bob is in a call “Phone_Call_1” and Jane is in a call “Phone_Call_1”.
- Rule 2: asserts Bob and Jane are in the same call.
- Rule 3 and 4: assert Alice, Jane, and Bob in the consultation “consultation_1”.
- Rule 5 asserts Web_NCAP is in the consultation “consultation_1”.

7 CONTEXT-BASED ACCESS CONTROL POLICY

This paper uses XACML as a policy language to support the concept of integrating contexts into an access control policy. XACML is being used for this type of extension because it is a standard access control policy language and it is extensible. There are two key concepts of extending XACML in this paper. First, an access control policy can be formed by using defined contexts and possibly their values (instances) in the semantic knowledge base directly. Second, contexts associated with a subject and resource from an access request are included.

In integrating contexts into an access control policy, this paper takes advantage of the use of attributes in XACML specifications since, according to the specifications, each subject and resource may contain multiple attribute values. Also, an access control decision can be based on some characteristic of the subject or resource other than its identity. In other words, policies can be formed based on subject or resource attributes. Hence, contexts are integrated into a policy as values of <AttributeID> tag and their instances as values of <AttributeValue>. This approach also applies for attributes of a subject and resource in an access request.

TABLE I : Partial semantic knowledge base in persistent database

Subject	Property	Object
Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#inConsultation:	Uv::http://www.w3.org/1999/02/22-rdf-syntax-ns#type:	Uv::http://www.w3.org/2002/07/owl#ObjectProperty:
Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#Alice:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#hasLocation:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#room_122:
Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#Alice:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#inConsultation:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#consultation_1:
Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#Jane:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#hasLocation:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#room_122:
Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#Jane:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#inConsultation:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#consultation_1:
Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#Bob:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#inCall:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#Phone_Call_1:
Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#Bob:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#inSameCall:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#Alice:
Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#Bob:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#inConsultation:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#consultation_1:
Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#Web_NCAP	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#hasLocation:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#room_122:
Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#Web_NCAP	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#inConsultation:	Uv::http://flame.cs.dal.ca/~dersingh/owl/ontology.owl#consultation_1:

Since contexts can be integrated as attribute values, it is also important to include contexts associated with a subject and resource in an access request from a PEP. This paper uses the mechanism of a query to the semantic knowledge base in order to search for contexts associated with the subject and resource in the knowledge base. Fig. 10 shows a data flow diagram in an extended XACML system. A context-aware access control decision and enforcement is now performed according to the following sequences:

0. This step is a preprocessing step. It starts by creating a context ontology at an Ontology Administration Point (OAP). The ontology is loaded to the context manager. The context manager as described previously monitors and analyzes domain information based on the ontology and stores the inferred knowledge associated with subjects, resources, and environments.
1. A Policy Administration Point (PAP) creates a XACML policy and provides it to the PDP. The PAP differs from the standard XACML in the sense that the PAP can lookup an ontology at the OAP in order to use context terms and form context-based access control (an example policy is shown in Fig.11).
2. An access requester sends an access request to a resource which get processed by a PEP
3. The PEP sends the request to the context handler which may include attributes of the subject, resource, action and environment
4. At run-time the context handler creates and constructs an XACML request and sends it to the PDP.
5. The PDP requests any additional subject, resource, and environment attributes from the context handler.
6. The context handler requests these additional attributes from a Policy Information Point (PIP).
7. The PIP obtains the requested attributes from the semantic subject, resource, and environment which means the attributes may be contexts of the situations such as “inConsultation”, “hasLocation”.
8. The PIP returns these attributes to the context handler.
9. Optionally, the context handler includes the resource content in the request.
10. The context handler sends the requested attributes (including context attributes) and, optionally, the content of the resource to the PDP. The PDP evaluates the access request against the policy.
11. The PDP returns the response decision to the context handler.
12. The context handler translates the response message back to the native response format and returns the response to the PEP.
13. The PEP fulfills the obligations.
14. (Not shown) if the access decision is “permit”, the PEP is allowed to access the resources; otherwise, it is denied.

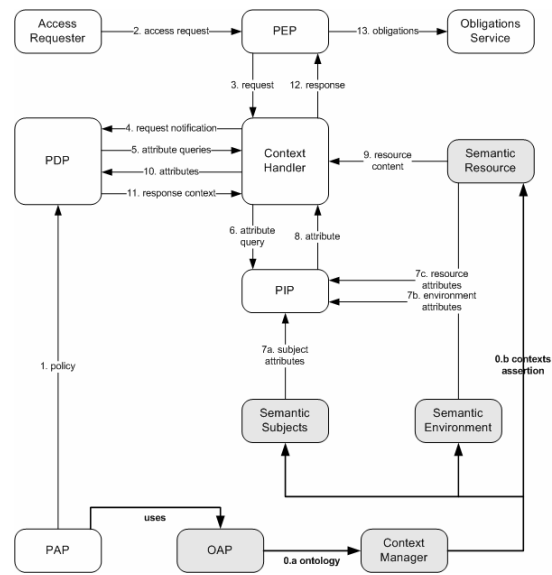


Figure 10: Data flow diagram

With the context modeling shown in Section 6, one can actually write down an access control policy based on the defined contexts directly. Fig. 11 shows an example of a partial XACML access control policy based on the context “inConsultation”. (Note that, for readability, Uniform Resource Identifier (URI) and Uniform Resource Name (URN) have been removed from the policy in Fig. 11 and request in Fig. 12. XACML specification [7] describes the use of URI and URN.)

```

<Rule RuleId="1" Effect="Permit">
  <Target>
    <Subjects> <AnySubject/> </Subjects>
    <Resources> <AnyResource/> </Resources>
    <Action> <AnyAction/> </Action>
  </Target>

  <Condition FunctionId="string-equal">
    <Apply FunctionId="string-one-and-only">
      <SubjectAttributeDesignator AttributeId="inConsultation"
        DataType="string"/>
    </Apply>
    <Apply FunctionId="string-one-and-only">
      <ResourceAttributeDesignator AttributeId="inConsultation"
        DataType="string"/>
    </Apply>
  </Condition>
</Rule>

```

Figure 11: Example of partial XACML context-based access control policy

This policy interprets our high-level policy by using the keyword “inConsultation”. And since the

previous rules already captured and asserted the participants of the consultation, this access policy simply uses the contexts and terms defined in the ontology and focuses on access management. The condition in the policy is to determine whether the subject (requester) and resource are in the same consultation or not by using the operator “equal” (FunctionId = “string-equal”). This approach helps network administrators in the sense that they can easily use contexts in policies without worrying about how to represent them semantically. Additionally, based on the policy shown in Fig. 11, the administrators do not have to instantiate an instance of a consultation room manually in the policy since it is provided in the semantic knowledge base.

```

<Subject>
  <Attribute AttributeId="subject-id" DataType="string">
    <AttributeValue>Bob:</AttributeValue>
  </Attribute>
  <Attribute AttributeId="inConsultation" DataType="string">
    <AttributeValue>consultation_1</AttributeValue>
  </Attribute>
</Subject>
<Resource>
  <Attribute AttributeId="resource-id" DataType="string">
    <AttributeValue>Web_NCAP</AttributeValue>
  </Attribute>
  <Attribute AttributeId="inConsultation" DataType="string">
    <AttributeValue>consultation_1</AttributeValue>
  </Attribute>
</Resource>

```

Figure 12: Example a partial XACML populated access request

However, the policy shown is evaluated against a request from a PEP. As mentioned in section IV, a request from a PEP is populated by the Context Handler. In other words, the context handler searches the semantic knowledge base to find contexts associated with the subject and resource. For example, the context handler receives a request which contains “Bob” as a subject, “Web_NCAP” as a resource, and “read” as an action. The context handler searches the semantic knowledge (Table 1) to find contexts associated with the subject and resource and populates the request. Fig. 8 shows a partial populated request from the context handler. In this populated request, both “Bob” and “Web_NCAP” are associated with “consultation_1” since the web NCAP server is located in a room where the consultation occurs and Bob is also participating the same consultation. Notice that this is a direct match in the database because of the pre-processing of the contexts.

8 PROTOTYPE IMPLEMENTATION

This paper validates the proposed approach with a proof of concept implementation. A prototype

composes of two parts: a context management system and an access control system. Both of them are implemented separately but are connected through a semantic knowledge base by which the context management system monitors and analyzes activities in the domain through sensing devices and provides semantic contexts in the semantic knowledge base while the access control system directly uses the contexts from the semantic knowledge base for access control policy evaluation.

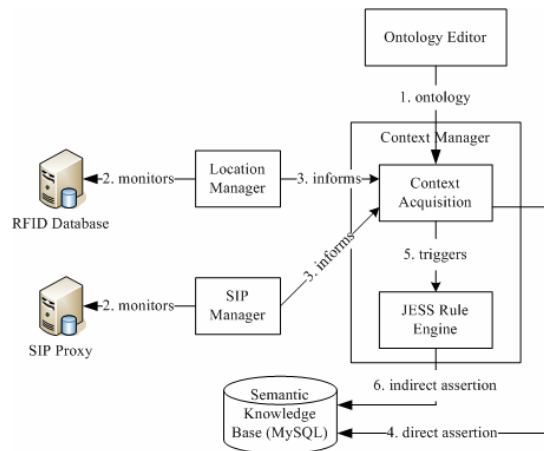


Figure 13: Context management system

Figure 13 shows components and data flow of the context management system. The data flow is as follows:

Step 1: The context ontology is loaded from the ontology editor, acts as an OAP, to a context acquisition process.

Step 2: A location manager and a SIP manager are processes that keep monitoring the environment. The location manager monitors RFID tags of the users and converts raw information from the RFID database to the format acceptable by the context acquisition. Similarly, the SIP manager monitors SIP calls in the domain and converts raw information to an appropriate format for the context acquisition.

Step 3: If there is a change in a user’s location or a SIP call activity, the location manager and the SIP manager inform the context acquisition respectively. The context acquisition, implemented by using Protégé [28] and Jena [29] APIs, analyzes the received information and properly associates the information with the context ontology. The context acquisition decides whether the information can be directly asserted to a semantic knowledgebase.

Step 4: A direct assertion is performed or,

Step 5: The information is passed on to a Jess rule engine [30] for further processing. The Jess rule engine infers the information using SWRL rules written as part of the context ontology.

Step 6: Results are asserted to the semantic knowledge base which is implemented as a MySQL

[31] relational database.

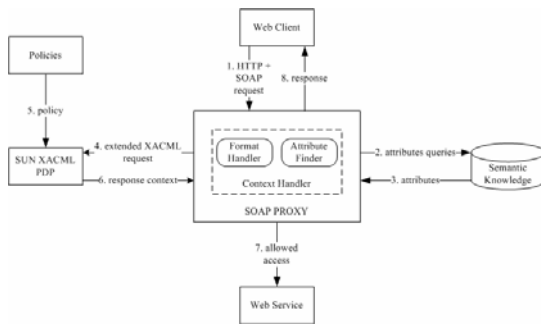


Figure 14: Access control system

The implementation of the context management system provides semantic contexts in the knowledge base. The implementation of the access control system focuses on using information stored in the semantic knowledge base. Fig. 14 shows the architecture and message sequences of the implemented access control system.

Referring back to the example scenario; assume that Bob, the expert, wants to access data from the ICU monitor located in an ICU room. Recall that for the implementation the ICU monitor here is the sensor web service application called Web NCAP that can read sensor data through a web service interface. At some point in the scenario, Bob was given a link to access the Web NCAP through his web browser, as shown in Fig. 15.

The message sequence is as follows:

Step 1: The web client at Bob's end sends an HTTP request to the web server. The HTTP message contains a SOAP body and Bob's SIP URI. Note that the SIP URI is required and used as an identity of the requester.

Step 2: At the web server there is a SOAP proxy that intercepts incoming requests to the service. After receiving the request, an attribute finder in the proxy performs queries to the semantic knowledge base in order to find attributes associated with subjects and resources. Attributes here are situational contexts such as "inConsultation".

Step 3: A format handler receives the attributes and translates the request to the XACML format understood by a XACML PDP.

Step 4: It then sends the extended XACML request to the XACML PDP.

Step 5: The PDP evaluates the request against a context-based access control policy.

Step 6: A response decision is sent back to the SOAP proxy.

Step 7: The proxy checks if the response is "permit". If so, the actual HTTP request is forwarded to the web service. If not an appropriate HTTP response is sent to the web client with an error message.

Notice that the format handler and attribute

finder act as a context handler in the extended XACML architecture.

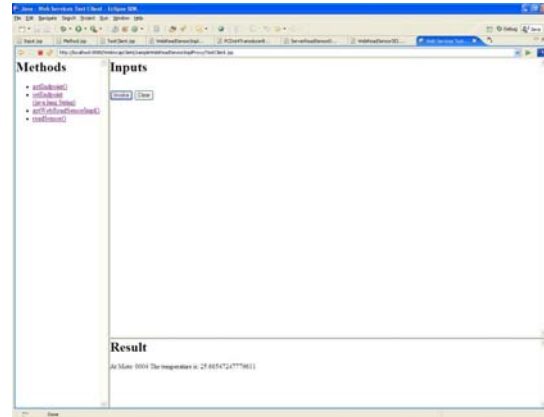


Figure 15: Web NCAP on Bob's web browser

9 CONCLUSION AND DISCUSSION

In autonomic computing and pervasive environments, the ultimate goal is to hide complexity from the user. This paper addresses this issue in the aspect of context-aware access control by demonstrating how contexts can be captured and represented semantically, and integrated into an access control policy. Also, this paper separates context management from access management so that administrators can focus on writing a policy or planning for appropriate actions in order to deal with anticipated threats or illegitimate accesses without worrying about how to capture or interpret domain knowledge. In addition, this approach reduces policy writing time since the administrators do not have to write all possible instantiations of the contexts as with non-semantic policy systems.

This paper also shows that access management can be less cumbersome for the user. For example, a remote user on one site can temporarily access a resource on another site without having to perform some complicated security access configuration. It is entirely based on contextual information such as users' activities. In other words, a dynamic group can be created based on the context of the situation and access requests are allowed if they are associated with that group or context.

Context assertion is the process of acquiring domain knowledge from sensors and rules. It is known that contexts of the situation change unpredictably over time. Once the state of the situation changes, it triggers the context acquisition process and the rule system to acquire the contexts of the current state. Current contexts have to replace or override previous contexts in the semantic knowledge base. Unfortunately this is not the case when using rules with OWL/RDF because of RDF's monotonicity assumption – once a triple is asserted it

is difficult to be retracted or changed. There are ways to resolve this problem that go beyond the scope of this paper and are currently being investigated by the authors.

Maintaining a consistent identity is also an important factor in these systems. In the paper the SIP URI is utilized and the authors are investigating the use of Identity 2.0 as an alternative.

The presented approach can also be applied to a different access control model such as RBAC. One can easily turn around and model roles in the same way as modeling contexts. It is also possible to have an access control policy that uses both roles and contexts in the policy. This basically will allow for the capture of relationships among roles.

In terms of policy languages, any policy language could have been used instead of XACML as long as it supports the notion of attributes. In fact because the semantic reasoning is kept in the context management no special policy language is required.

ACKNOWLEDGEMENT

This work was supported in part by the National Science and Engineering Council under Grant 262045-04.

10 REFERENCES

- [1] N. Chase, "An autonomic computing roadmap", <http://www.ibm.com/developerworks/library/ac-roadmap>, 2004.
- [2] B. Foote, and J. Yoder, "Big Ball of Mud", in *Pattern Languages of Program Design 4*, ed. N. Harrison, B. Foote, H. Rohnert, Addison-Wesley, 2000.
- [3] P. Lin, A. MacArthur, and J. Leaney, "Defining Autonomic Computing: A Software Engineering Perspective", *Proc. Australian Software Engineering Conference*, pp. 88-97, 2005.
- [4] J. O. Kephart, D. M. Chess. "The Vision of Autonomic Computing." *Computer*, Jan., 2003, pp41-50.
- [5] M. Weiser, "Creating the Invisible Interface", *Symposium on User Interface Software and Technology*, New York, NY, ACM Press, 1994.
- [6] S. Dobson et. al., "A Survey of Autonomic Communications", *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, pp. 223-259, December 2006.
- [7] T. Moses, "eXtensible Access Control Markup Language (XACML) version 2.0," 2005, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.
- [8] H. Cheng, T Finin, and A. Joshi, "An Ontology for Context-Aware Pervasive Computing Environments," *Proc. IJCAI Workshop on Ontologies and Distributed Systems*, IJCAI, 2003, <http://www.cs.vu.nl/~heiner/IJCAI-03/Papers/Chen.pdf>.
- [9] A. Corradi, R. Montanari, and D. Tibaldi, "Context-based Access Control for Ubiquitous Service Provisioning", pp. 444-451, 28th Annual International Computer Software and Applications Conference (COMPSAC'04), 2004.
- [10] L. Kagal, "A Policy-Based Approach to Governing Autonomous Behavior in Distributed Environments", *Dissertation*, 2004.
- [11] L. Kagal, and T. Berners-Lee, "Rein: Where policies meet rules in the semantic web," *Technical report*, MIT, 2005.
- [12] L. Kagal, T. Finin, and A. Joshi, "A Policy Language for a Pervasive Computing Environment", In *IEEE 4th International Workshop on Policies for distributed Systems and Networks*, 2003.
- [13] A. Patwardhan, V. Korolev, L. Kagal, and A. Joshi, "Enforcing Policies in Pervasive Environments", *International Conference on Mobile and Ubiquitous Systems: Networking and Services*, August 2004.
- [14] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and L. Lott., "KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement", In *Proc. 4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'03)*, page 93, 2003.
- [15] E. Damiani, S. De Capitani di Vimercati, C. Fugazza, and P. Samarati, "Extending Policy Languages to the Semantic Web", *International Conference on Web Engineering (ICWE2004)*, *Lecture Notes in Computer Science*, pp. 330-343, July 2004.
- [16] T. Priebe, W. Dobmeier, and N. Kamprath, "Supporting Attribute-based Access Control with Ontologies," *First International Conference on Availability, Reliability and Security (ARES'06)*, pp. 465-472, 2006.
- [17] A. Toninelli, R. Montanari, and L. Kagal, O. Lassila, "A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments", *International Semantic Web Conference*, pp. 473-486, 2006.
- [18] V. C. Hu, D. F. Ferraiolo, and D. R. Kuhn, "Assessment of Access Control Systems", *National Institute of Standards and Technology (NIST), Technology Administration, U.S. Department of Commerce, Interagency Report 7316*, September 2006.
- [19] A. Dersingh, R. Liscano, and A. Jost, "Bridging the Policy Gap in Pervasive Access Control: A Semantic Web Approach," *4th International Workshop on Managing Ubiquitous Communications and Services*, Munich,

- Germany, May 2007.
- [20] G. Antoniou, and F. V. Harmelen, “A Semantic Web Primer”, The MIT Press Cambridge, Massachusetts London, England, 2004.
- [21] J. Heflin, “OWL Web Ontology Language Use Cases and Requirements”, <http://www.w3.org/TR/webont-req/>, February 2004.
- [22] W. Chunkun, “Policy-based Network Management”, *Proceeding of IEEE International Conference on Communication Technology*, vol 1, pp. 101-105, August 2000.
- [23] S. Waldbusser, J. Saperia, and T. Hongal, “Policy Based Management MIB”, IETF, RFC 4011, March 2005.
- [24] N. Damianou, N. Dulay, E. Lupu, M. Sloman, “The PONDER Policy Specification Language”, In Proc. International Workshop of Policies for Distributed Systems and Networks (Policy 2001). Bristol, UK, January 2001. LNCS 1995: 18-39, Springer-Verlag (2001).
- [25] A. Corradi, R. Montanari, and D. Tibaldi, “Context-based Access Control for Ubiquitous Service Provisioning”, pp. 444-451, 28th Annual International Computer Software and Applications Conference (COMPSAC'04), 2004.
- [26] D. L. McGuinness, and F. van Harmelen, “OWL Web Ontology Language Overview”, <http://www.w3.org/TR/owl-features/>, February 2004.
- [27] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosz, M. Dean: SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission (21 May 2004).
- [28] Protégé Editor and API, <http://protege.stanford.edu> Accessed July 2007.
- [29] Jena API for Java, <http://jena.sourceforge.net> Access July 2007.
- [30] Jess Rule Engine, <http://herzberg.ca.sandia.gov/jess> Accessed July 2007.
- [31] MySQL database, <http://www.mysql.com>. Accessed July 2007.
- [32] E. F. Sadok and R. Liscano, “A Web Services Framework for 1451 Sensor Networks”, Proceedings of the 2005 IEEE Instrumentation and Measurement Technology Conference, 2005. (IMTC 2005) Ottawa, ON Canada May 17-19, 2005.
- [33] C. McGregor and B. Kneale, Simulated Neonatal Intensive Care Units to Support Neonatologist International Mobility”, Proceedings of the Third IASTED International Conference of Telehealth (Telehealth 2007), Montral, Canada.