Rule-based and Ontology-based Policies: Toward a Hybrid Approach to Control Agents in Pervasive Environments

Alessandra Toninelli[1], Jeffrey M. Bradshaw[2], Lalana Kagal[3], Rebecca Montanari[1]

[1]Dipartimento di Elettronica, Informatica e Sistemistica
Università di Bologna
Viale Risorgimento, 2 - 40136 Bologna - Italy
{atoninelli, rmontanari}@deis.unibo.it

[2] Florida Institute for Human and Machine Cognition (IHMC)
40 S. Alcaniz Street, Pensacola, FL 32502, USA
jbradshaw@ihmc.us

[3]MIT CSAIL
32 Vassar Street, Boston, MA, USA
lkagal@csail.mit.edu

Abstract. *Policies are being increasingly used for controlling the behavior of complex multi-agent systems. The use of policies allows administrators to regulate agent behavior without changing source code or requiring the consent or cooperation of the agents being governed. However, policy-based control can sometimes encounter difficulties when applied to agents that act in pervasive environments characterized by frequent and unpredictable changes. In such cases, we cannot always specify policies a priori to handle any operative run time situation, but instead require continuous adjustments to allow agents to behave in a contextually appropriate manner. To address these issues, some policy approaches for governing agents in pervasive environments specify policies in a way that is both context-based and semantically-rich. Two approaches have been used in recent research: an ontology-based approach that relies heavily on the expressive features of Description Logic (DL) languages, and a rule-based approach that encodes policies as Logic Programming (LP) rules. The aim of this paper is to analyze the emerging directions for the specification of semantically-rich context-based policies, highlighting their advantages and drawbacks. Based on our analysis we describe a hybrid approach that exploits the expressive capabilities of both DL and LP approaches.*

1. Introduction

The multi-agent paradigm offers a promising software engineering approach for the development of applications in complex environments [1]. By their ability to operate autonomously without constant human supervision, agents can perform tasks that would be impractical or impossible using traditional software techniques [2]. However, this autonomy, if unchecked, has also the potential of causing severe damage if agents are poorly designed, buggy, or malicious.

Explicit policies can help in dynamically regulating the behavior of agents and in maintaining an adequate level of security, predictability, and responsiveness to human control. Policies provide the dynamic bounds within which an agent is permitted to function autonomously and limit the possibility of unwanted events occurring during operations. By changing policies, agent behavior can be continuously adjusted to accommodate variations in externally imposed constraints and environmental conditions without modifying the agent code or requiring the cooperation of the agents being governed [3].

Until recently, policies have been primarily exploited to govern complex distributed systems within traditional computing environments that rely on a relatively fixed set of resources, users, and services. However, with the Internet becoming ubiquitous, researchers started to investigate how to develop adequate policy-based techniques for controlling agent behavior within pervasive environments [4]. The dynamicity, unpredictability and heterogeneity of pervasive environments complicate the design of policy languages and techniques for agent control. Resources are not pre-determined, interacting agents are not always known a priori and, if agents roam across different network localities, they have different resource visibility and availability, depending on their location and on other context-dependent information, such as local security policies and resource state. In this setting, agents need to be provided with a semantically clear and interoperable description of the context where they execute and need to acquire, reason about and negotiate the policies that rule their behavior in each new context, so that they can decide whether to adhere or not. In addition, policies for controlling agent behavior cannot always be specified beforehand to handle any operative run-time situation, but may require a high rate of dynamic and continuous adjustments to allow agents to act in any execution context in the most suitable way and to accommodate context changes.

To address these issues, some policy approaches for pervasive environments are starting to emerge that share common design principles [5, 6, 7]. A significant design concept that guides these approaches is the use of contextual information for driving policy specifications. Since context is a prime quality of pervasive environments, it should explicitly appear in policy specifications. In fact, in pervasive scenarios it is almost impossible to know the identities or roles of all agents that are likely to interact and request services in advance. Instead of defining identity- or role-based policies, administrators may more easily define the conditions for making resources available and

for allowing or denying agents resource visibility and access, according to the context of their operating conditions.

Another important design principle is the adoption of a semantically-rich representations for policy definition. Semantically-rich representations permit both structure and properties of the elements of a pervasive system and the management operations themselves (e.g., policies) to be described at a high level of abstraction, thus enabling policy conflict detection and harmonization.

Recent research efforts in the area of semantically-rich context-based approaches to policy representation follow one of two possible directions. *Ontology-based* approaches rely largely on the expressive features of Description Logic languages, such as OWL [7], to classify contexts and policies, thus enabling deductive inferences and static policy conflict resolution to be performed. In contrast, *rule-based* approaches take the perspective of Logic Programming to encode rules in a clear, logic-like way. Moreover, a rule-based approach facilitates the straightforward mapping of policies to lower level enforcement mechanisms thanks to its concise and understandable syntax.

The scope of this paper is to analyze the emerging directions for the specification of semantically-rich context-based policies, highlighting their advantages and drawbacks. Based on our analysis we describe a hybrid approach that exploits the expressive capabilities of both approaches. On the one hand, it should rely on an ontology-based approach to enable policy classification, comparison and static conflict resolution. On the other hand, it should be able to reap the benefits of a rule-based approach, thus enabling the efficient enforcement of policies defined over dynamically determined values.

The structure of the paper follows. Section 2 outlines some fundamental requirements for policy languages to enable the specification of semantic context-based policies. Section 3 analyzes how some relevant well-known approaches to semantic policy representation, i.e., KAoS and Rei, deal with the aforementioned requirements. The comparison allows us to discuss, in Section 4, a possible direction toward the integration of ontology-based and rule-based policy approaches in order to exploit their full advantages. Conclusions and future work are presented in Section 5.

2. Novel Requirements for Semantic Context-based Policies

The control of agent behavior in pervasive scenarios raises novel requirements for the design of policy languages and policy run-time environments. In pervasive scenarios, users, on behalf of whom agents act, typically move from one environment to another, thus determining continuous variations in their physical position and in their execution context, including the set of entities and resources they may be able to interact with. Moreover, users can access the network using various devices, e.g., laptops, PDAs or mobile phones, which exhibit different capabilities in terms of resources and computational abilities. As it is not possible to exactly predict all the interactions an entity may be involved in and the kind of resources it may wish to have access to, policy-based

control cannot rely on any precise knowledge about the subjects/events/actions that need to be governed.

To deal with such environmental characteristics, recent research efforts propose to adopt semantically-rich representations to express policy and domain knowledge. The adoption of Semantic Web languages to specify and manage policies in pervasive computing scenarios brings several advantages. In fact, semantically-rich representations ensure that there is a common understanding between previously unknown entities about their capabilities, the current execution context and the actions they are permitted or obliged to perform. Moreover, modeling policies at a high level of abstraction simplifies their description and improves the analyzability of the system. Semantic Web languages also enable expressive querying and automated reasoning about policy representation.

Another emerging direction suggests that, in order to deal with the dynamic context changes that are typical of pervasive applications, it may be advantageous to build policies directly over context conditions, i.e., to consider context as a primary element in the specification of policies [8]. Context is a complex notion that has many definitions. Here we consider context as any information that is useful to characterize the state or the activity of an entity, e.g., its location or its characteristics, and any useful information about the world in which this entity operates, e.g., date and time. In pervasive environments, where client users are typically unknown and where context operating conditions frequently change even unpredictably, the specification of context-based policies, instead of traditional subject-or role-based ones, allows to control the behavior of entities without having to foresee all the possible interactions that an entity may have with other entities and resources.

The adoption of a semantic context-based policy approach to control pervasive systems requires the definition of a policy model that can precisely identify the basic types of policies required to control agents, can specify how to express and represent context and related policies in a semantically expressive form, and how to enforce them. In this paper we focus particularly on specification rather than on enforcement issues. To this extent, we consider the following as basic requirements for a semantic-based policy language:

- the ability to model and represent the contexts in which agents operate and to which policies are associated, at a high level of abstraction.
- the ability to define what actions are permitted or forbidden to do on resources in specific contexts (*authorizations* or *permission/prohibition* policies);
- the ability to define the actions that must be performed on resources in specific contexts (*obligations*).

The aim of this paper is not to provide a general survey of the state-of-the-art in context-based policy representation, but to carefully analyze how some relevant semantic policy approaches deal with the specification of semantic context-based policies. We first present KAoS, followed by Rei, both of which were originally designed for governing agent behavior and that represent, respectively, significant examples of ontology-based and rule-based policy languages. Then, from this analysis, we derive some suggestions toward the design of a hybrid policy approach.

3. Semantic Approaches to Context-based Policy Specification

To illustrate the expressive capabilities of the two considered policy frameworks, i.e., KAoS and Rei, let us consider a usage scenario that is likely to become more and more usual in the next future. Let us consider the case of a traveler, Alice, who is waiting at the airport for her flight to leave. The airport is equipped with several 802.11b hot spots providing travelers with wireless connectivity for their portable devices, e.g., laptops, PDA and mobile phones. Airline companies may also provide additional services and resources to travelers, such as the possibility to print documents stored on their devices using public printers that are placed in various areas of the airport. Moreover, while waiting to board, users may wish to share files with other users, by exploiting the wireless connectivity available at the airport. Since she has to wait a couple of hours for her plane to leave, Alice starts to work on some documents she has on her laptop. Therefore, she may wish to access the printer that is available in the waiting area around her boarding gate to print the documents she needs. In addition, as she likes jazz music very much, she would like to exchange music files with other travelers waiting in the airport hall. These activities need to be regulated by appropriate policies. In particular, the following policies governing adequate access to services and resource sharing may apply. We will use these policies as a running policy example throughout the rest of the paper.

LocationSharing Policy
Users that are currently co-located with the owner of the policy, i.e., with her device, are authorized to access the shared files stored on the owner device.

This policy may be instantiated and enforced by Alice to share her music files with co-located travelers in a secure way, depending on current context conditions.

PrinterAccess Policy
Travelers that are flying with a company of the Sky Team group, and are currently located in the airport area including gate from 31 to 57 are authorized to access the printer.

This policy may be enforced by the provider of a printing service that is offered to travelers flying with the Sky Team alliance in some areas of the airport. The enforcement of this authorization should ensure that travelers having proper rights are enabled to access the service. Furthermore, if the default behavior of the system states that everything that is not explicitly permitted is prohibited, this policy also prevents unauthorized travelers from accessing the service.

In the following sections we first analyze KAoS, and subsequently Rei, showing how they deal with the specification of the previously described policies.

KAoS

KAoS is a framework that provides policy and domain management services for agent and other distributed computing platforms [9, 10, 17]. It has been deployed in a wide variety of multi-agent and distributed computing applications. KAoS policy services allow for the specification, management, conflict resolution and enforcement of policies within agent domains. KPAT, a powerful graphical user interface, allows non-specialists to specify and analyze complex policies without having to master the complexity of OWL.
KAoS adopts an ontology-based approach to semantic policy specification. In fact, policies are mainly represented in OWL [7] as ontologies. The KAoS policy ontologies distinguish between authorizations and obligations. In KAoS, a policy constrains the actions that an agent is allowed or obliged to perform in a given context. In particular, each policy controls a well-defined action, whose subject, target and other context conditions are defined as property restrictions on the action type. Figure 1a shows an example of KAoS authorization, which represents the *PrinterAccess* policy previously described. The property *performedBy* is used to define the class to which the actor must belong for the policy to be satisfied.

```
<owl:Class rdf:ID="SkyTeamGate31-57PrinterAccessAction">
<owl:intersectionOf rdf:parseType="Collection">
 <owl:intersectionOf rdf:parseType="Collection">
 <rdfsowl:Class rdf:about="&action;AccessAction"/>
  <owl:Restriction>
      <owl:onProperty rdf:resource="&action;performedBy"/>
      <owl:allValuesFrom rdf:resource="#SkyTeamCustomer"/>
 </owl:Restriction>
  <owl:Restriction>
      <owl:onProperty rdf:resource="&action;accessedEntity"/>
      <owl:allValuesFrom rdf:resource="#Printer31-57"/>
  </owl:Restriction>
</owl:intersectionOf>
 </owl:Class>


    < policy:PosAuthorizationPolicy rdf:ID=" SkyTeamGate31-57PrinterAccess">
      < policy:controls rdf:resource="# SkyTeamGate31-57PrinterAccessAction"/>
      <policy:hasSiteOfEnforcement rdf:resource="&some-ontology;TargetSite"/>
      <policy:hasPriority>10</policy:hasPriority>
    </policy:PosAutihorizationPolicy>
```

a)

```
    <owl:Class rdf:ID="SkyTeamCustomer">
    <rdfs:subClassOf rdf:resource="&some-ontology;Customer"/>
     <rdfs:subClassOf>
       <owl:Restriction>
            <owl:onProperty rdf:resource="&some-ontology;firm"/>
            <owl:allValuesFrom rdf:resource="&some-ontology;SkyTeamAlliance"/>
      </owl:Restriction>
```

```
    </rdfs:subClassOf>
```

b)

Fig. 1. KAoS policy examples.

In KAoS, context conditions that constrain a policy may be specified through the definition of appropriate classes defined via property restrictions. In particular, two main properties, i.e., the *hasDataContext* and the *hasObjectContext* properties, and their sub-properties are used to characterize the action context. Some sub-properties are defined in the KAoS ontology, like for instance the ones defining the actor (*performedBy*), the time and the target resource (*accessedEntity*) of an action, while others may be created within domain-specific ontologies. Figure 1b shows the definition of a class, namely *SkyTeamCustomer*, which represents all the individuals that are flying with a company belonging to the Sky Team alliance. This class is defined as a subclass of the *Customer* class, having the *affiliation* property restricted to the Sky Team.

As these examples show, KAoS is based on an *ontological* approach to policy specification, which exploits OWL, i.e., description logic, features to describe and specify policies and context conditions. In fact, contexts and related policies are expressed as ontologies. Therefore, KAoS is able to classify and reason about both domain and policy specification basing on ontological subsumption, and to detect policy conflicts statically, i.e., at policy definition time.

However, a pure OWL approach encounters some difficulties with regard to the definition of some kinds of policies—specifically those requiring the definition of *variables*. For instance, by relying purely on OWL, we could not define policies such as the *FileSharing* policy, which defines constraints over property values that refer to statically unknown values, e.g., the policy owner location. Other examples include policies that contain parametric constraints, which are assigned a value only at deployment or run time. For this reason, KAoS developers have introduced role-value maps as OWL extensions and implementing them within the Java Theorem Prover, used by KAoS [11, 17]. The adoption of role value maps, description logic-based concept constructors that were originally introduced in the KL-ONE system [12], allows KAoS to specify constraints between property values expressed in OWL terms, and to define policy sets, i.e., groups of policies that share a common definition but can be singularly instantiated with different parameters. The proposed extensions effectively add sufficient expressive flexibility to KAoS to represent the policies discussed in this paper. However, non-experienced users may have difficulties in writing and understanding these policies without the help of the KPAT graphical user interface.

Rei

Rei is a policy framework that permits to specify, analyze and reason about declarative policies defined as norms of behavior [4, 6]. Rei adopts a rule-based approach to specify

semantic policies. Rei policies restrict domain actions that an entity can/must perform on resources in the environment, allowing policies to be developed as contextually constrained deontic concepts, i.e., right, prohibition, obligation and dispensation. The first version of Rei (Rei 1.0) is defined entirely in first order logic with logical specifications for introducing domain knowledge [13]. The current version of Rei (Rei 2.0), that we analyze in this paper, adopts OWL-Lite to specify policies and can reason over any domain knowledge expressed in either RDF or OWL [4].

A policy basically consists of a list of rules and a context that is used to define the policy domain. Rules are expressed as OWL properties of the policy. In particular, the *policy:grants* property is used to associate a deontic object with a policy either directly or via a *policy:Granting*. Figure 2 shows the Rei 2.0 policy specification of the *LocationSharing* policy. In order to specify context conditions, one or more constraints must be defined. A constraint, which may be simple or boolean, i.e., the boolean combination of a pair of simple constraints, defines a set of actors or a set of actions that fulfill a certain property. A simple constraint, as shown in Figure 2b, is modeled as a triple consisting of a subject, a predicate and an object, which defines the value of the property for the entity, following a pattern that is typical of logical languages like Prolog.

A constraint can be associated to a policy at three different levels. The first possibility is to impose a constraint within the definition of a deontic object, by means of the *deontic:constraint* property, as shown in Figure 2c. In this case, the constraint can be expressed over the actor, the action to be controlled or over generic environmental states, e.g., the time of the day. Additional constraints can be imposed within the Granting specification over the entity the granting is made to, the deontic object the granting is made over and, again, over generic environmental states. Finally, it is possible to express a set of constraints directly within the policy definition through the *policy:context* property. These constraints are generically defined as conditions over attributes of entities in the policy domain.

```
<policy:Policy rdf:ID="FileAccessPolicy">
    <policy:actor rdf:resource="#requester"/>
    <policy:grants rdf:resource="#Perm_FileAccess"/>
</policy:Policy>
<policy:Policy rdf:ID=”FileSharingPolicy”>
  ...
</policy:Policy>
```

a)

```
<constraint:SimpleConstraint rdf:ID="LocationOfUser">
     <constraint:subject rdf:resource="&some-ontology;user"/>
     <constraint:predicate rdf:resource="&some-ontology;location"/>
     <constraint:object rdf:resource="#user-location"/>
</constraint:SimpleConstraint>

<constraint:SimpleConstraint rdf:ID="CoLocatedWithUser">
     <constraint:subject rdf:resource="#requester"/>
     <constraint:predicate rdf:resource="&some-ontology;location"/>
```

```
     <constraint:object rdf:resource="#user-location"/>
</constraint:SimpleConstraint>

<constraint:And rdf:ID="Constraint_CoLocated">
     <constraint:first rdf:resource="#LocationOfUser"/>
     <constraint:second rdf:resource="#CoLocatedWithUser"/>
</constraint:And>
```

b)

```
<deontic:Permission rdf:ID="Perm_FileAccess">
    <deontic:actor rdf:resource="#requester"/>
    <deontic:action rdf:resource="&some-ontology;AccessToSharedFiles"/>
    <deontic:constraint rdf:resource="#Constraint_CoLocated"/>
</deontic:Permission>
```

c)

Fig. 2. Rei policy examples.

Rei 2.0 uses OWL-Lite for the specification of policies and of domain-specific knowledge. Though represented in OWL-Lite, Rei still allows the definition of variables that are used as placeholders as in Prolog. In fact, as shown in Figure 2b, the definition of constraints follows the typical pattern of rule-based programming languages, like Prolog, i.e., defining a variable and the required value of that variable for the constraint to be satisfied. In this way, Rei overcomes one of the major limitations of the OWL language, and more generally of description logics. i.e., the inability to define variables. For example, as shown in Figure 2, Rei allows developers to express a policy stating that a user is allowed to access the shared files of another user if they are located in the same area, whereas pure OWL would not allow for the definition of the "same as" concept. Therefore, Rei's rule-based approach enables the definition of policies that refer to a dynamically determined value, thus providing greater expressiveness and flexibility to policy specification.

On the other hand, the choice of expressing Rei rules similarly to declarative logic programs prevents it from exploiting the full potential of the OWL language. In fact, Rei rules knowledge is treated separately from OWL ontology knowledge due to its different syntactical form. OWL inference is essentially considered as an oracle, i.e., the Rei policy engine treats inferences from OWL axioms as a virtual fact base. Hence, Rei rules cannot be exploited in the reasoning process that infers new conclusions from the OWL existing ontologies, which means that the Rei engine is able to reason about domain-specific knowledge, but not about policy specification. As a main consequence of this limitation, Rei policy statements cannot be classified by means of ontological reasoning. Therefore, in order to classify policies, the variables in the rules need to be instantiated, which reduces to a constraint satisfiability problem. Let us consider, for example the previously described *PrinterAccess* policy. Unlike KAoS, Rei does not allow for a policy disclosure process that retrieves policies controlling a specific *type* of action. Hence, the user willing to use the printer could only try to access it and see what the Rei engine has answered, with regard to this particular access. For the same reason, Rei cannot statically detect

conflicts, like KAoS does, but it can only discover them with respect to a particular situation.

4. Toward a Hybrid Approach to Semantic Policy Specification?

The management of context and related policies is a demanding task and requires the appropriate description of context and subsequent policies. Our analysis of current approaches to semantic context-based policy specification has outlined two main research directions, which move from two opposite sides.

On one side, a purely ontology-based approach exploits description logic, e.g., OWL, to describe contexts and associated policies at a high level of abstraction, in a form that allows their classification and comparison. This feature is essential, for instance, in order to detect conflicts between policies before they are actually enforced, thus granting interoperability among entities belonging to different domains that adopt different policies. In fact, by means of a preliminary analysis of policy typologies holding in different domains, the required behaviors of each domain can be compared and harmonized, if needed, avoiding the cost of failures due to conflicts arising in the enforcement phase. Another interesting application of an ontology-based approach lies in the possibility of exploiting policy description to facilitate negotiation in policy disclosure. As an entity may wish to interact with potentially untrusted entities, negotiating policy disclosure may help interacting parties in the effort of reaching an agreement about their mutual behavior without imposing too heavy limitations to their privacy.

On the other side, a rule-based approach relies on the features of logic programming languages, e.g., Prolog, to enable evaluation and reasoning about concrete context and policy instances. In fact, from the enforcement point of view, policy rules can be considered as "instructions" to be executed provided that their activating conditions, i.e., contexts, are evaluated to be true. This perspective suggests that contexts and related policies should be expressed in a clear, concise and expressive way to facilitate their evaluation and enactment, similarly to the code of a programming language that needs to be compiled or interpreted. For example, the language should allow for the definition of policies over dynamically determined constraints, including run time variables, as this is a crucial expressive feature that most programming languages offer.

KAoS and Rei represent intermediate approaches between the two opposite approaches previously described. KAoS was originally based only on description logic, provided by the OWL language features, but current features aim at overcoming the intrinsic limitations of OWL as a description logic-based language, i.e., the inability to allow variable-based reasoning [11, 17]. Rei, the first version of which was strongly oriented to a declarative logic programming approach, has recently moved from a Prolog-like syntax to an OWL encoding that permits ontological reasoning over domain knowledge (but not over policy rules), mainly to solve the extensibility problem of Rei 1.0.

We claim that a policy framework for pervasive computing systems should be able to provide support to context modeling and evaluating with different levels of granularity and flexibility. In particular, we suggest the possibility of an integrated approach that exploits both description logic (DL) and logic programming (LP). At a higher level of granularity and abstraction, DL should be exploited to classify contexts and related policies, thus allowing static conflict resolution and favoring gradual policy disclosure between interacting parties. At a more operational level, LP should be used to encode rules in a clear and expressive fashion that may also facilitate their enforcement.

Let us consider, for instance, the *LocationSharing* policy. This policy can be described using description logic through the definition of the ontological concepts of location context and co-location, thus enabling classification, comparison and static conflict detection with other policies that are related with the same concepts. On the other side, logic programming can be used to encode the rule that effectively enforces the policy, namely: if user Y is located in the same place as user X, then user Y is allowed to access user X's shared files.

Let us illustrate with an example how a policy framework could benefit from such a hybrid approach to policy specification. Let us suppose that Bob is waiting to check-in at the airport and wishes to share some music files with other travelers at the airport. In order not to waste battery, he would like to avoid a random approach where he just tries to access other users devices to share files with them, without knowing in advance if the access will be permitted or denied. To this extent, before attempting to send or receive files from his portable device, he asks other users to disclose their public access control policies. Let us suppose that the *LocationSharing* policy is in force and active on Alice laptop. Upon receiving Bob request for policy disclosure, she retrieves and sends the policies that controls the "file sharing" action type, i.e., the *LocationSharing* policy. At this point, Bob can statically check for conflicts between Alice's policy and his own policies controlling the same type of action, i.e., file sharing. If, for instance, Bob has enforced a policy to control file sharing that does not include any location context, then he can deduce that there is no conflict between his own policy and Alice's. It is worth noting that policy disclosure and conflict detection is enabled by the ontology-based definition of policies. On the other hand, when Bob actually tries to access Alice shared files, the access control policy is enforced on the basis of its rule-based definition, by evaluating the current value of variables, i.e., Alice and Bob current location. Let us note that, due to the dynamic nature of the policy whose evaluation can be made only at access time, it may still be possible that Bob is not allowed to access Alice files because of his current location. However, thank to the preliminary policy disclosure phase, Bob is able to decide whether he agrees to adhere to a policy that imposes some conditions on user location. If, for instance, Bob is waiting at the check-in desk and he already knows that his location will not change in the next hour because there is a very long queue, then he may decide to choose another user that does not impose any condition on location.

It is worth stating that an integrated approach like the one we have described would require the establishment of a semantic and inferential correspondence between DL and LP. This is a complex issue, which nonetheless may be addressed, as demonstrated, for

instance, by recent work. For example, the approach described in [15] could represent a valid guideline toward a viable integration process. According to the authors, the idea was on the one hand to enable to "build rules on top of ontologies", i.e., enable the rule knowledge base to have access to DL ontological definitions for vocabulary primitives, and on the other hand to enable to "build ontologies over rules", i.e., enable ontological definitions to be supplemented by rules, or imported into DL from rules. Let us note that Rei seems to have taken the first approach, as in its latest version it allows to specify policy rules using policy and domain ontologies. KAoS, by means of appropriate extensions of the OWL language, is aiming at supplementing its ontological specification of policies with rules. In particular, the authors of [15] propose a mapping between DL and LP, based on the consideration that, under appropriate restrictions, both logics can be considered as restricted sets of first order logic. As a final consideration, we believe that the way toward interoperation between rules and ontologies could be further explored to achieve more powerful and flexible expressive means for the specification of context-based policies.

5. Conclusions and Future Work

The specification of semantically-rich context-based policies to regulate agent behavior in pervasive environments is a complex task that requires appropriate representations to describe both context information relevant for policy specification and the policies themselves. Our analysis of current approaches to semantically-rich context-based policy specification has described two main research directions that are moving toward the middle from two opposite sides, i.e., an ontology-oriented approach, based on description logic features, and a rule-oriented approach, based on logic programming. The paper proposes a hybrid approach to policy specification that allows better handling of the highly dynamic, uncertain and heterogeneous conditions that are typical of the pervasive environments where agents operate. This paper has analyzed the problem and the issues to be solved in developing such a hybrid policy approach. Further investigation is needed to conduct more formal and thorough analyses of existing and proposed systems in order to understand their strengths and weaknesses, and to propose the basis for new research and development. Along this direction, stimulating ideas and results can come from the investigation of existing proposals, such as SWSL [16], which describes the attempt to combine first-order logic with rule based languages to specify the Semantic Web Services ontologies as well as individual Web services.

References

1. Jennings, N., "An agent-based approach for building complex software systems"., Communications of the ACM, 44(4), pp. 35-41, 2001.

2. Bradshaw, J. M. (Ed.). Software Agents. Cambridge, MA: The AAAI Press/The MIT Press, 1997.
3. Bradshaw, J. M., Jung, H., Kulkarni, S., & Taysom, W. "Dimensions of adjustable autonomy and mixed-initiative interaction". In M. Klusch, G. Weiss, & M. Rovatsos (Ed.), Computational Autonomy, Springer-Verlag, Berlin, Germany, 2004.
4. Kagal, L.: "A Policy Based Approach to Governing Autonomous Behavior in Distributed Environments". Dissertation submitted to the Faculty of the Graduate School of the University of Maryland for the degree of Doctor of Philosophy, Baltimore County, USA, 2004.
5. Tonti, G., Bradshaw, J. M., Jeffers, R., Montanari, R., Suri, N., Uszok, A.: "Semantic Web languages for policy representation and reasoning: A comparison of KAoS, Rei, and Ponder", Proc. of the Second International Semantic Web Conference (ISWC2003), LNCS, Vol. 2870. Springer-Verlag, Berlin, pp. 419-437, Sanibel Island, Florida, USA, October 2003.
6. Kagal, L., Finin, T., Joshi, A.: "A Policy Language for Pervasive Computing Environment" In: Proc. of IEEE Fourth International Workshop on Policy (Policy 2003). Lake Como, Italy, pp. 63-76, IEEE Computer Society Press 4-6 June 2003.
7. Van Harmelen, F., et al.: "OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004", http://www.w3.org/TR/owl-ref/.
8. Montanari, R., Toninelli, A., Bradshaw, J.M.: "Context-Based Security Management for Multi-Agent Systems", To be published In: Proc. of the Second IEEE Symposium on Multi-Agent Security and Survivability, IEEE Press, Philadelphia, USA, 30-31 August 2005.
9. Bradshaw, J. M., Uszok, A., Jeffers, R., Suri, N., Hayes, P., Burstein, M. H., Acquisti, A., Benyo, B., Breedy, M. R., Carvalho, M., Diller, D., Johnson, M., Kulkarni, S., Lott, J., Sierhuis, M., & Van Hoof, R. (2003). Representation and reasoning for DAML-based policy and domain services in KAoS and Nomads. *Proceedings of the Autonomous Agents and Multi-Agent Systems Conference (AAMAS 2003).* 14-18 July, Melbourne, Australia. New York, NY: ACM Press, pp. 835-842
10. Uszok, A., et al.: "KAoS policy management for semantic web services". IEEE Intelligent Systems, 19(4), p. 32-41, 2004.
11. Moreau, L., Bradshaw, J., Breedy, M., Bunch, L., Johnson, M., Kulkarni S., Lott J., Suri N., Uszok A.: "Behavioural Specification of Grid Services with the KAoS Policy Language", Proc. of the Cluster Computing and Grid 2005. Cardiff, UK, 9-12 May 2005.
12. Schmidt-Schauss, M.: "Subsumption in KL-ONE is undecidable", In: Proc. of the First Intl Conference on the Principles of Knowledge Representation and Reasoning (KR 1989), Morgan Kaufmann: Los Altos, 1989.
13. Kagal, L.: Rei: "A Policy Language for the Me-Centric Project", HP Labs Technical Report, HPL-2002-270, 2002.
14. N. Damianou, et al., "The Ponder Policy Specification Language," Proc. 2nd Int'l Workshop Policies for Distributed Systems and Networks, LNCS 1995, Springer-Verlag, pp. 18-38, 2001.
15. Grosof, B.N., Horrocks I., Volz, R., and Decker, S.: "Description Logic Programs: Combining Logic Programs with Description Logic", In: Proc. of WWW 2003, 20-24 May 2003, Budapest, Hungary, 2003.
16. Battle S., et al., Semantic Web Service Language (SWSL), Version 1.0. http://www.daml.org/services/swsf/1.0/swsl/ (2005).
17. Uszok, A., Bradshaw, J. M., Jeffers, R., Tate, A. & Dalton, J. (2004). Applying KAoS services to ensure policy compliance for semantic web services workflow composition and enactment. In S. A. McIlraith, D. Plexousakis, F. van Harmelen (Eds.), *The Semantic Web—ISWC 2004,* Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan, November 7-11, LNCS 3298, Berlin: Springer, pp. 425-440.