A Policy Enforcement Framework for Ubiquitous Computing Applications

Ioannis Panagiotopoulos, Lambrini Seremeti, Achilles Kameas
School of Science and Technology
Hellenic Open University
Patras, Greece
giannis.panagiotopoulos@gmail.com, seremeti@cti.gr, kameas@eap.gr

***Abstract*—Future ubiquitous computing environments integrate the services of everyday objects equipped with tiny processors and sensors into distributed applications. These smart devices can communicate with each other and also explore their environment. In order for the applications to function properly, policies need to be defined, which determine ways that they can be used, protected, changed, etc. A policy can be considered as a set of rules, specified by users, which are usually applied by a policy manager. In this paper we proposed an alternative approach, which supports the adoption of policies directly by the applications without the need of an enforcing policy manager. Two everyday scenarios are used as examples that demonstrate the validity of the approach.**

Ontology; Policy; Ubiquitous computing; Ontology alignment; Protégé; Alignment API; Jena

I. Introduction

Nowadays, tiny embedded processors are found in everyday items including mobile phones, TV's, cooking appliances, washing machines etc, while many of them can be connected to the Internet. In the near future, networking and communication capabilities will be integrated at low cost, giving them the ability to collaborate with each other and also explore their environment [1]. Ubiquitous computing applications are defined, which orchestrate the services offered by these devices; then, the smart devices are regarded as resources of a Next Generation Ambient Intelligence Environment (NGAIE) [2]. One of the main problems detected for NGAIEs is the heterogeneity of the constituent resources.

One proposed approach to deal with the heterogeneity of the ubiquitous computing environments resources is to use ontologies to model the state, knowledge and services of the resources and the policies that apply to the environment or the applications they are found in. Ontologies permit the clear definition and explicit specification of the basic concepts of a concrete field, thus facilitating communication and interoperation between heterogeneous entities (such as humans, services or software agents). Thus ontologies are considered as an important tool for the successful implementation of ubiquitous computing applications.

Policies are also used in ubiquitous computing systems to determine ways that these applications can be used, protected, changed, etc. In general, policies are considered as sets of high-level rules which describe the way a system or an application should behave under different circumstances. There are many types of policies, such as access control policies, which control the permissions of the users over a resource, privacy policies, which define the privacy boundaries and protection mechanisms, interaction policies, which specify how a user can interact with an application, etc. In this paper, we concentrate on the realization of policies that control the operation of smart devices within a specific ubiquitous computing application. We describe a framework that enables a user to model the features of the application and use them to define his/her own policies, which can then directly be applied to the resources that compose the application he/she owns or uses. The innovation of the proposed approach is that policies can be applied directly by the application manager, without the need (and cost) of extra software (i.e. a dedicated policy manager).

The rest of the paper is structured in the following way. In section II is discussed the work related to policy management in ubiquitous computing applications and the justification of the proposed methodology. Section III presents the two scenarios used to demonstrate the proposed framework. In section IV, the basic steps of the proposed policy specification methodology are developed and in section V these steps are implemented to the abovementioned scenarios. Lastly, section VI discusses the conclusions of the presented research.

II. Related Work

Several efforts have addressed the issue of policy management in ubiquitous computing environments.

In CASA [3] the authors proposed a secure architecture for context-aware environments. They focus on defining a security middleware to provide flexible access control and policy management. A security management service is responsible for managing and storing policies defined by the domain administrator. The policy manager provides the interface for the definition of policies, which are encoded in XML and stored in a policy repository.

KAoS [4] is a policy language with support for the specification, management and enforcement of policies. The policies are represented through ontologies in DAML (http://www.daml.org). KAoS services and tools provide software components, people, resources and other entities the capability to be semantically described and structured into organizations of domains. While initially oriented to the requirements of software agent applications, the services have been adapted to web services environments also. It has a graphical tool called KAoS Policy Administration Tool (KPAT) that helps users in the specification, revision and application of policies. Guards (software agents) are responsible for policy enforcement within the computational environment, while Enforcers are the mechanism by which Guards ensure compliance with authorization policies.

Rei [5] is another language for expressing policies. It is a highly expressive policy specification language well suited for describing security policies in pervasive environments. Rei defines a policy as a set of rules describing concepts like permission, prohibition, obligation and dispensation over all possible actions within the environment. Rei's policy engine reasons over policies described in Rei language, and uses the policies and the domain knowledge to make security decisions about access right and obligations.

Patwardhan et al. [6] proposed a security infrastructure that uses Rei to define security policies and use policy enforcement mechanisms on the mobile devices in order to eliminate the possible threats posed to the device. According to the proposed architecture, a policy engine reasons over the policies described in Rei and grants or denies access to requests made by individuals in the domain. Then the policy server presents the policy engine the state information of the device in question (location, identifier and person in possession) and consults the engine to create a new policy certificate with the granted requests. The policy manager is responsible for retrieving policies from the policy server, while the policy enforcer is the access mediator located on the device. It is responsible for enforcing the current policy that has been verified to be issued by a trusted resource.

Jiang et al. [7], proposed a middleware which provides the services to the user in order to define security policies that reflect dynamic context. They defined three kind of policies used in the policy management service: authorization policy, delegation policy and obligation policy. The policy manager provides the interface for the administrator to define the policies. The policies are encoded in XML and are stored in a policy repository.

FOCALE [8] is an autonomic architecture for managing context-aware services, such as those required by ubiquitous computing applications. This architecture uses a context-aware policy model [9] that can generate ontologies to govern behavior. This policy model is connected to another context model, so that policies that use resources or services can sense the context changes. When the context changes, causes policies to change and thus the functionality offered by the entity (e.g. device). The context manager locates the changes of the state related with the context and the policy management system selects a set of policies that should be loaded and activated based on the current context.

Almuhaideb et al. [10] proposed a ubiquitous access model to provide the mobile users with a flexible authentication method to access foreign network services, for example when they travel. The design of this model is based on two tokens, an authentication token and an authorization token. Trust and negotiation are two essential components for the cooperation between entities in ubiquitous computing environments. The engaging entities use policies to govern trust and negotiation. A policy manager contains the rules (policies) defined by each party according to their interests, before the negotiation between them. The policies defined are trust, authorization, identification and policies concerning features like quality of service and security.

Some of these approaches focus on access control and authentication for ubiquitous computing applications but all of them make use of a policy manager as a mediator between the user and the system. The proposed approach on the other hand, offers a more generic schema where users define their own preferences for the function of the devices. Moreover all the above approaches use a policy manager to control the definition, storing and enforcement of all the policies defined by the user/system administrator. In the proposed framework policies are stored on the devices and not in a policy repository. It is a user-centric approach because user defines and manages his/her preferences by applying them directly to the devices, without a middleware manager.

III. Scenarios

To illustrate how policies could be applied without a specific policy component, two everyday scenarios will be used:

A. 1st Scenario

Suki is in his smart home. He wants to wash a few clothing items but he is not sure about the way the clothes can be washed properly. So his is checking the internet through his PDA about basic washing instructions, which subsequently, he encodes as policies directly to his smart washing machine through an interface which his machine is equipped with. These policies mainly describe incompatibilities between color and texture, which have to be taken into account, so as not to damage the clothes during washing. A washing policy for example can indicate that "colored and white clothing items cannot be washed together" or "a woolen clothing item cannot be washed together with a cotton clothing item". The washing machine can identify the color and the texture of the clothes via embedded RFID tags, and can decides whether it is safe to wash a specific combination of clothing items.

B. 2nd Scenario

This time Suki wants to iron the clothes that were washed in the previous scenario, but he is not sure about the right use of the iron, especially regarding temperature. He consults again his PDA for ironing instructions and uses them to define ironing policies to his smart iron. The iron through an RFID reader recognizes the texture of the clothes and notifies Suki of policy conflicts related to the ironing service. An example of such policy is that "a clothing item made of wool should not be ironed".

IV. Policy Enforcement Methodology

In this section, is presented step by step the development of the proposed approach and the final model that provides the framework for the policy application.

A. *Modeling devices and policies*

During the first step of the methodology ontologies have been used to model the devices, the items and the policies for the two scenarios. For the first scenario, three ontologies have been created: the *WashingMachine* ontology, the *Clothing* ontology and the *WashingPolicy* ontology. The first ontology models the washing machine device, the second the clothing items and the third the policies which are expressed as rules in the ontology.

Respectively, three ontologies have been used for the second scenario: the *Iron* ontology, the *IroningPolicy* ontology and the *Clothing* ontology from the previous scenario.

B. *Ontology alignment*

To achieve homogeneity and interoperability between the three different ontologies of each scenario we applied ontology alignment among pairs of ontologies for each scenario separately. An alignment is a set of correspondences between entities (e.g. classes, properties, individuals) occurring in the ontologies. All the alignments are exported as ontologies and are used for ontology merging in the next step.

C. *Ontology merging*

During this step we use the results of the previous steps in order to create the final model. This model consists of the merged ontologies from steps *A* and *B*. In this way common knowledge is being shared between the different ontologies.

D. *Final model querying*

The final step of the methodology concerns the application of the model. In order to extract information from the final model and to test the function of the policies defined, a number of queries are being applied to the model. It is worth noting that the queries are applied directly to the device ontologies, which can respond stating whether they support a service or not. Note that during the construction of the ontologies in the first step, no value has been assigned to a service provided by a device. This is exactly the information the devices infer from their connection with the policy ontologies during the alignment process of the second step.

V. Policy Application Framework

In this section, we describe the application of the steps presented in the previous section in the two scenarios we presented in section III, using a number of specific tools for each step. The aim is to demonstrate that the proposed framework can be realized within the context of ubiquitous computing applications. For our purposed, we model ubiquitous computing applications composed of resources, each of which comes with its own local ontology; we have coined the metaphor of "activity spheres" to model such self-contained applications [11]. The global state of an activity sphere is encoded in the Sphere Ontology, which results from the alignment of all the ontologies that represent the sphere's resources, agents, policies, etc. Thus, instead of using a policy manager to apply the washing and ironing policies, we encode them as ontologies, which can be accessed by the Activity Sphere Manager.

For the construction of the ontologies, the tool Protégé [12] has been used. In Fig. 1 is depicted the *Clothing* ontology which is common for both scenarios.

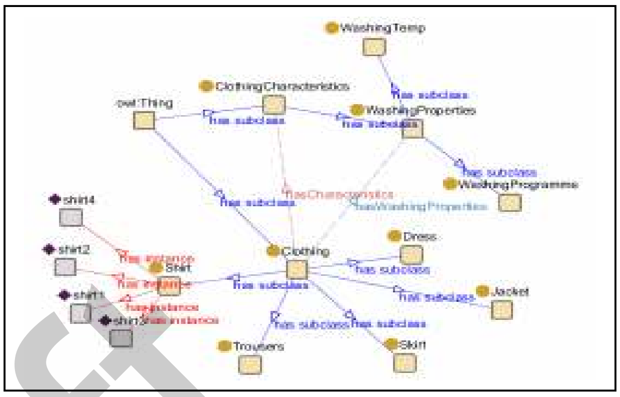
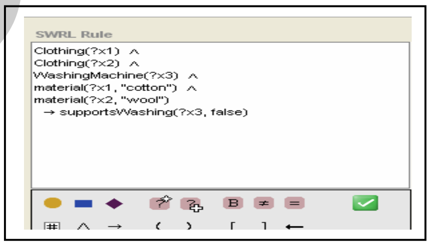


Figure 1. The Clothing ontology

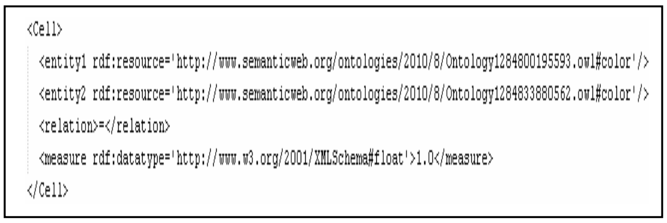
Policies have been expressed as SWRL [13] rules in the policy ontology for both scenarios. In Fig. 2 we can see the second washing policy of the first scenario, provided as example in section III.


SWRL Rule

```
Clothing(?x1) ^
Clothing(?x2) ^
WashingMachine(?x3) ^
material(?x1, "cotton") ^
material(?x2, "wool")
  → supportsWashing(?x3, false)
```

Figure 2. Policy expressed in SWRL language

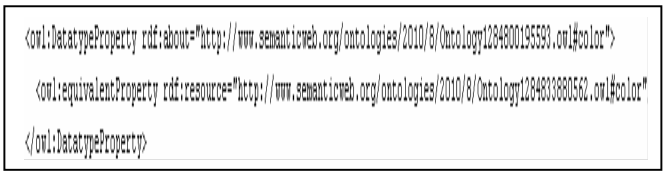
For the alignment process the Alignment API [14] has been used. The API offers a number of predefined algorithms to provide the alignment between two ontologies. In the current implementation the *StringDistAlignment* algorithm has been used to compute the substring distance on the entity names. Moreover a threshold (0.9) has been used in selecting more accurate correspondences. In Fig. 3 we can see an example of alignment output in RDF [15]:



```
<Cell>
  <entity1 rdf:resource='http://www.semanticweb.org/ontologies/2010/8/Ontology1284800195593.owl#color'/>
  <entity2 rdf:resource='http://www.semanticweb.org/ontologies/2010/8/Ontology1284833880562.owl#color'/>
  <relation>=</relation>
  <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</measure>
</Cell>
```

Figure 3. Sample of RDF code

The above code indicates that the data property *color* in the *Clothing* ontology is equivalent with the data property *color* in the *WashingPolicy* ontology. In this case OWL [16] ontologies were merged, so the output format was changed into a set of OWL axioms. The API provides the notion of visitors of the alignment cells which are used to render the alignments. In the present implementation the *OWLAxiomsRendererVisitor* has been used, which generates an ontology, merging both aligned ontologies and comprising OWL axioms for expressing the subsumption and equivalence relations. In Fig. 4 appears the above code (Fig. 3) as an OWL axiom:

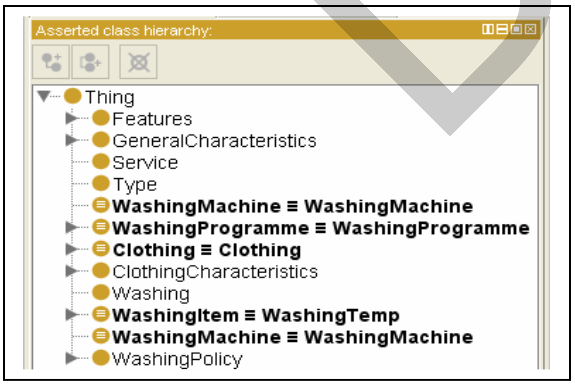


```
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/ontologies/2010/8/Ontology1284800195593.owl#color">
  <owl:equivalentProperty rdf:resource="http://www.semanticweb.org/ontologies/2010/8/Ontology1284833880562.owl#color"
</owl:DatatypeProperty>
```

Figure 4. Example of OWL axiom

The next step was to export all the set of axioms as OWL ontologies so that they could be merged with the ontologies created for modeling the scenarios of the previous section. This can be achieved through an option in the Protégé. Thus the result is three new ontologies for each scenario which were merged with the scenarios ontologies.

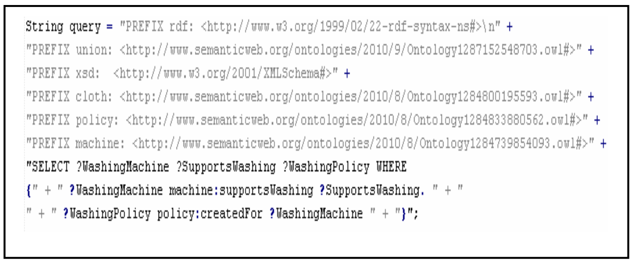
In order to merge the ontologies Jena [17] has been used. Jena provides a programming environment and a query engine for RDF and OWL ontologies. It also provides inference support so as to extract additional information from these models. For this purpose Jena offers a variety of reasoners which can be plugged to the model. In our case we used the Pellet reasoner [18] because it offers support for SWRL rules, which as previously mentioned were used to describe the policies for each activity in the two scenarios. Fig. 5 shows the basic classes of the merged model and the equivalences between them.



```
Thing
  Features
  GeneralCharacteristics
  Service
  Type
  WashingMachine ≡ WashingMachine
  WashingProgramme ≡ WashingProgramme
  Clothing ≡ Clothing
  ClothingCharacteristics
  Washing
  WashingItem ≡ WashingTemp
  WashingMachine ≡ WashingMachine
  WashingPolicy
```

Figure 5. Basic classes of the final model

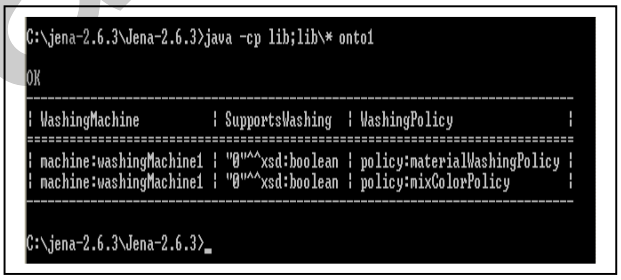
Jena provides the ability to apply queries in SPARQL [19]. Below we show an example query. In order to apply the query, four instances in the *Clothing* ontology of the first scenario were created: a white shirt and a colored one, a shirt made of wool and a shirt made of cotton. Also in the *WashingMachine* ontology a washing machine instance was created to represent the device. In this query, shown in Fig. 6, the washing machine is asked if it supports the mixed color washing (whites and colored items) or clothing items of different texture together and which is the policy controlling these functions. It is expected that the washing machine will not support the washing for the above clothing items with these characteristics and due to the policies pre-defined.



```
String query = "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>\n" +
"PREFIX union: <http://www.semanticweb.org/ontologies/2010/9/Ontology1287152548703.owl#>" +
"PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>" +
"PREFIX cloth: <http://www.semanticweb.org/ontologies/2010/8/Ontology1284800195593.owl#>" +
"PREFIX policy: <http://www.semanticweb.org/ontologies/2010/8/Ontology1284833880562.owl#>" +
"PREFIX machine: <http://www.semanticweb.org/ontologies/2010/8/Ontology1284739854093.owl#>" +
"SELECT ?WashingMachine ?SupportsWashing ?WashingPolicy WHERE
{" + " ?WashingMachine machine:supportsWashing ?SupportsWashing. " + "
" + " ?WashingPolicy policy:createdFor ?WashingMachine " + "}";
```

Figure 6. SPARQL query

In Fig. 7 is depicted the result of the query. As it was expected the washing machine doesn't support the washing of a white and a color clothing item put together and also the washing of items of different fabric at the same time. In the first column appears the washing machine, the second provides the answer (represented as Boolean) and in the third column the policy that controls the function. As expected the answer is negative. Otherwise, if the clothing items had other characteristics (e.g. all the same color) the answer would be positive and the washing machine would support the washing service.



```
C:\jena-2.6.3\Jena-2.6.3>java -cp lib;lib\* onto1

OK
-----------------------------------------------------------------------------
| WashingMachine          | SupportsWashing  | WashingPolicy                  |
=============================================================================
| machine:washingMachine1 | "0"^^xsd:boolean | policy:materialWashingPolicy |
| machine:washingMachine1 | "0"^^xsd:boolean | policy:mixColorPolicy        |
-----------------------------------------------------------------------------


C:\jena-2.6.3\Jena-2.6.3>_
```

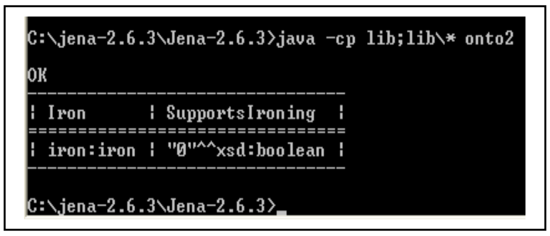
Figure 7. Result of the SPARQL query

The activity sphere composed by the Washing Machine and the Clothing items is managed by a dedicated Sphere Manager. An Ontology Manager is responsible for creating the Sphere Ontology by merging the ontologies and the various policies using Jena. The Sphere Manager, in order to realize specific tasks related to washing clothes, sends queries to the Ontology Manager. In turn, the Ontology Manager applies the queries to the Sphere Ontology and provides the results to the Sphere Manager. In our approach, the results of these queries already integrate the related policies, thus a separate policy enforcing component is not necessary.

For the second scenario the exact same methodology can be followed. So, at this point we are not going to repeat all the steps but will present only the results of the query application for the second model. As before, a dedicated Sphere Manager is realized that manages the Ironing sphere and a dedicated

Ontology Manager manages the Ironing sphere ontology, which integrates the ironing policy.

This time the *Iron* ontology is asked if supports the ironing service for clothing item made of wool. The result is shown in Fig. 8.



```
C:\jena-2.6.3\Jena-2.6.3>java -cp lib;lib\* onto2

OK
--------------------------------
| Iron      | SupportsIroning  |
================================
| iron:iron | "0"^^xsd:boolean |
--------------------------------

C:\jena-2.6.3\Jena-2.6.3>
```

Figure 8. Result of the SPARQL query

VI. CONCLUSIONS

This paper proposed a framework for policy definition and application in ubiquitous computing applications that are composed from services offered by heterogeneous smart resources. Policies are defined as rules that are applied on the resource ontologies. This framework can be used for any type of policy including security, privacy, interaction etc. The main advantage is that no specific policy enforcement component is required, because policy ontologies are merged with other resource ontologies and they can be accessed by any ontology manager. The proposed framework has been applied in the context of the ATRACO project [20] and currently is limited to the types of applications that were developed in the project. We are now working on creating a generic schema for any type of policy created by the users.

ACKNOWLEDGMENT

The research leading to the above results has been partially conducted within project ATRACO and was partially funded from the European Community 7th Framework Programme (FP7/2007-2013) under grant agreement No. 216837.

REFERENCES

[1] J. Ahola, "Ambient Intelligence," ERCIM News, (47), October 2001.

[2] T. Heinroth, A. Kameas, G. Pruvost, L. Seremeti, Y. Bellik and W. Minker, "Human-Computer Interaction in Next Generation Ambient Intelligent Environments". Intelligent Decision Technologies, special issue on Knowledge-based environments and services in HCI, 5(1), 2011, IOS Press, to appear.

[3] M. J. Covingtony, P. Fogla, Z. Zhan, "A Context-Aware Security Architecture for Emerging Applications," Procceding of the Annual Computer Security Applications Conference, December 2002.

[4] A. Uszok et al., "KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement," Proceedings of the fourth IEEE International Workshop on Policies for Distributed Systems and Networks, 2003.

[5] L. Kagal, T. Finin, an A. Joshi, "A Policy Language for A Pervasive Computing Environment," Proceedings of the fourth IEEE International Workshop on Policies for Distributed Systems and Networks, 2003.

[6] A. Patwardhan, V. Korolev, L. Kagal, and A. Joshi, "Enforcing policies in Pervasive Environments," Proceedings of the first Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004.

[7] Z. Jiang, K. x.K. Lee, S. Kim, H. Bae, S. Kim, S. Kang, "Design of a Security Management Middleware in Ubiquitous Computing Environments," Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies, pp. 306 – 308, 2005.

[8] J. Strassner, S. van der Meer, B. Jennings, M. P. De Leon, "An Autonomic Architecture to Manage Ubiquitous Computing Networking and Applications," Proceedings of the first International Conference on Ubiquitous and Future Networks, July 2009.

[9] J. Strassner, J. de Souza, D. Raymer, S. Samudrala, S. Davy, K. Barrett, "The Design of a New Policy Model to Support Ontology-Driven Reasoning for Autonomic Networking," Journal of Network and Systems Management, vol. 17, pp. 114-125, 2007.

[10] A. M. Almuhaideb, M. A. Alhabeeb, P. D. Le, B. Srinivasan, "Flexible Authentication Techique for Ubiquitous Wireless Communication using Passport and Visa Tokens," Journal of Telecommunications, vol. 1, pp. 1-10, March 2010.

[11] L. Seremeti, C. Goumopoulos and A. Kameas, "Ontology-based modeling of dynamic ubiquitous computing applications as evolving activity spheres". Pervasive and Mobile Computing, 5(5), 2009, Elsevier Science, pp 574-591.

[12] H. Knublauch, M. Horridge, M. Musen, A. Rector, R. Stevens, N. Drummond, P. Lord, N. F. Noy, J. Seidenberg, H. Wang, "The Protégé OWL experience," Workshop on OWL: Experiences and Directions. In fourth International Semantic Web Conference, 2005. Available at: http://protege.stanford.edu/

[13] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, "SWRL: A Semantic Web Rule Language combing OWL and RuleML," W3C Submission, May 2004. Available at: http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/

[14] Alignmet API. Available at: http://alignapi.gforge.inria.fr/

[15] Resource Description Framework. Available at: http://www.w3.org/RDF/

[16] Ontology Web Language. Available at: http://www.w3.org/TR/owl-features/

[17] Jena – A Semantic Web Framework for Java. Available at: http://jena.sourceforge.net/

[18] Pellet: OWL Reasoner for Java. Available at: http://clarkparsia.com/pellet/

[19] SPARQL Query Language in RDF. Available at: http://www.w3.org/TR/rdf-sparql-query/

[20] ATRACO Project. Available at: http://www.uni-ulm.de/in/atraco