GRID METADATA LIFETIME CONTROL IN ACTON

Wei Xing, Rizos Sakellariou, Oscar Corcho, Carole Goble
School of Computer Science
University of Manchester
United Kingdom
wxing@cs.man.ac.uk
rizos@cs.man.ac.uk
ocorcho@cs.man.ac.uk
carole@cs.man.ac.uk

Marios D. Dikaiakos
Department of Computer Science
University of Cyprus, Cyprus
mdd@cs.ucy.ac.cy

Abstract In the Semantic Grid, metadata, as first class citizens, should be maintained up-to-date in a cost-effective manner. This includes maximising the automation of different aspects of the metadata lifecycle, managing the evolution and change of metadata in distributed contexts, and synchronising adequately the evolution of all these related entities. In this paper, we introduce a semantic model and its operations which is designed for supporting dynamic metadata management in Active Ontology (ActOn), a semantic information integration approach for highly dynamic information sources. Finally, we illustrate the ActOn-based metadata lifetime control by EGEE examples.

Keywords: Grid, Semantic Model, Active Ontology, Semantic Grid Information Integration, EGEE

1. Introduction

Metadata in most Grid applications and middleware systems are managed either in an ad-hoc manner or their processing is hard-wired inside the middleware code. The arbitrary expression and use of knowledge causes Grid middleware to be more prone to syntactic changes, less (if at all) interoperable, more dependent on extensive human effort for deployment configuration and maintenance, and less shareable. This seriously hampers the progress towards flexible, adaptable and interoperable Grid infrastructures [6]. The Semantic Grid proposes to address this problem by constructing so-called "**metadata buses**" for Grid infrastructures [9, 12, 5]. One role of *Semantics* is the creation of machine understandable metadata that will allow the automation of important tasks such as service discovery and dynamic composition of workflows. And subsequently, the management of semantics/metadata of Grid services, workflows, and datasets on/in Grids.

Grid infrastructures comprise many time-sensitive resources and services with characteristics that change frequently, at different time-scales. For example, the usage of CPU resources and the status of job queues may change in minutes, the stability of services may change in hours, the metadata about membership to a virtual organisation (VO) may change in days. Consequently, one of the main challenges for the management of Grid-metadata arises from the dynamic nature of the Grid, that is its *dynamicity*.

To address this challenge, we designed and developed Active Ontology (ActOn) [12]. ActOn is an ontology-based information model, which extends the OWL syntax with a set of classes and properties designed to enable: (i) the lifetime control of dynamic metadata and (ii) the specification of queries that retrieve dynamic metadata from external information sources; these queries are embedded into OWL class/instance descriptions [10]. The main idea behind ActOn is that it allows the association of time-variant elements of an OWL class with embedded queries that manage the provenance of these elements. ActOn extends the architectural ideas and techniques that have been proposed in the context of ontology-based information integration, by supporting the dynamic selection of information sources selection according to changing information needs and the state of available information sources. In order to have up-to-date resource metadata without making continuous update requests, we create a metadata cache, which works with an on-demand-update policy, so that only active metadata are aggregated in an efficient and economic manner [13].

In this paper, we describe the ActOn semantic information model and illustrate how it can be used to manage dynamic metadata in a large-scale distributed system. We first describe the syntax and semantics of the ActOn semantic information model, and the operations of the ActOn system. Then we describe ActOn's support for lifetime-control and for on-demand updates of dynamic

metadata. We also give an EGEE Grid example to illustrate how the semantic model can be used in a real Grid system [8].

The remainder of the paper is organised as follows: Section 2 gives a high level overview of the ActOn approach. Section 3 presents the ActOn semantic model, focusing on its syntax and semantic definition. Section 4 illustrates how the ActOn semantic model works with lifetime control mechanisms and updating-on-demanding policy. Finally, Section 5 provides conclusions, and describes open issues and our planned future work.

2. A High Level Overview of Active Ontology

Active Ontology (ActOn) is designed to support and manage dynamicity in large-scale distributed systems, using Semantic Web technology. Dynamicity is a key feature of many large-scale distributed systems, such as Grids; for example, the performance capacity of distributed Grid resources and the availability of software services change dynamically. Consequently, an ontology that describes a distributed system, like the Core Grid Ontology which represents a Grid infrastructure [14], should be able to capture the dynamicity of that system. Nevertheless, OWL ontologies are used to encode static information that does not change over a long period of time, such as concepts of a domain and the relationships among these concepts [4]. The standard OWL DL [11]does not provide the expressive power required to represent time-sensitive properties of concepts and relationships. Hence, the representation of dynamic information in an OWL ontology remains a challenge. To address this challenge, we introduced ActOn, an "Active" Ontology that manages dynamic changes of the instances of a large-scale distributed system ontology.

The idea of ActOn is simple: ActOn allows the dynamic parts of an OWL class definition to be described by embedded queries, in a way similar to the AXML proposal [16]. Unlike the traditional assignment of fixed data-type values to properties of OWL instances, ActOn can assign query objects to time-variant properties. The query is embedded into an OWL class/instance definition, so that it can be dynamically executed. Hence, the values of the time-sensitive properties can be fetched dynamically in order to update the instances according to the changes that take place.

ActOn provides the means to embed query functions into the definition of an ontology in order to generate instances of its classes automatically and then update them dynamically. In principle, an ActOn query can be presented in any kind of query language even as just a wrapper with a simple UNIX script. This makes ActOn more flexible and rigorous for maintaining the information about a dynamic distributed system, such as Grids.

3. The ActOn semantic model

The ActOn semantic model is composed of a domain ontology (DO), an information sources ontology (ISO) and the ActOn Linker. The definition of the Active Ontology (ActOn) in BNF syntax is as follows:

```
Active Ontology ::= 'Ontology(' [ domain ] ')'
                  | 'Ontology('[information sources]')'
                  | 'ActOn('{Linker}')'
```

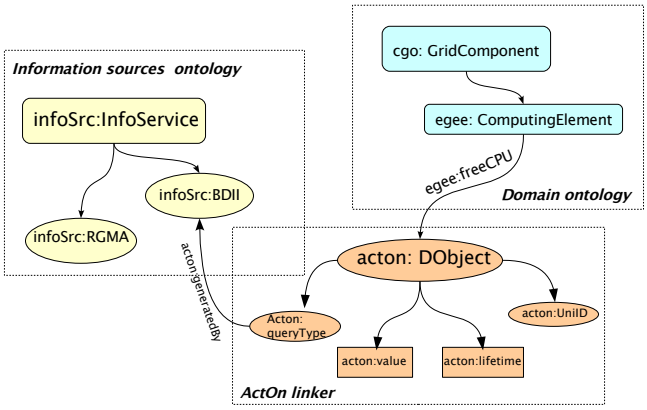

Figure 1. The ActOn Semantic Model: Domain & Information Source Ontologies and the ActOn Linker

The domain ontology (DO) provides a framework for representing and capturing the configuration and state of a distributed system. It describes the entities of a distributed system and their relationships as a set of concepts, relations, and individuals, which are represented, respectively, as OWL classes, properties and instances. Entities represented by the DO include hardware resources, applications, software components, services, etc. For example, in the DO of the EGEE Grid, the *ComputingElement* class represents the concept of a Grid component that provides access to computing resources (CPUs). The details of this definition can be found in [14]. We will use this class as an example to illustrate the ActOn syntax and semantics later. Different distributed systems may have different ontologies, which can be loaded into ActOn. This makes ActOn an architecture-independent approach for building semantic-based Grid information services [7].

The ISO defines the classes and properties of *information sources*, that is, network-enabled entities providing information about the configuration and state of DO elements. Different types of information sources are described as subclasses of a generic *InfoSrc* class. For example, BDII is an information service of the EGEE Grid infrastructure that provides information about available hardware resources such as computing power (e.g., CPUs). Therefore, in ISO, we define the *BDII* class as a subclass of *InfoSrc*. The details of the EGEE ISO ontology can be found in [2]. Specific information-source entities are, in essence, instances of the architecture-specific subclasses of *InfoSrc*. These instances can be created either manually, by an ontology editor like Protégé [15], or automatically, by the ActOn instances generator service (described in [12]).

The role of the Linker is enable the management of dynamically changing metadata expressed in OWL. To this end, the Linker links the two ontologies (DO and ISO) together, as shown in Figure 1. The Linker holds and manages ActOn DObject's, which contain lifetime information about metadata and the embedded queries for retrieving dynamic metadata updates. The key idea of ActOn is that it embeds a query into an OWL entity in order to adapt dynamically the part of the entity that changes with time. The ActOn Linker generates a list of queries for execution by managing the list of DObject's that are stored in its database. We will describe the DObject in detail in the next section.

We separate the domain ontology and the information sources ontology in ActOn for two reasons: (i) To make ActOn flexible and extensible. Thanks to this separation, ActOn can easily retrieve metadata from a variety of external information sources, which can change dynamically and which are typically heterogeneous, geographically distributed, and under different administrative domains. (ii) To Support dynamic management. Dynamic changes will be both in DO and ISO, however in different aspects and frequency. The separation will make the dynamic changes in one ontology not affect the other. For instance, the change of the information sources ontology, such as to add or remove an information source, will not affect the definition of the domain ontology as the definition of the instance in a domain ontology is not fixed with a specific instance in the information ontology.

3.1 The ActOn Dynamic Object

In order to represent and manage OWL properties that change dynamically (*dynamic properties*), we introduce the DObject class. Every *dynamic property* of a DO class instance is assigned a DObject instance, instead of a statically given value. An instance of the DObject class captures the information about the time-stamp of the actual metadata value wrapped inside the DObject instance at hand and the embedded query used to retrieve that value from some information source (see Figure 1). For example, the *ComputerElement* class

has a dynamic property freeCPU; the value of that property is an instance of the DObject class, which contains a query for retrieving the value of the property freeCPU, the integer value of freeCPU, and the lifetime of that value.

By an instance of DObject, the value of the dynamic property of instances of a OWL class can actually be represented as a triple set, which are an embedded query, a value, and the lifetime of the value. So that the value can be updated based on its lifetime by executing the query dynamically.

As the formalism definition of the DObject in DL in (1) below, an ActOn DObject is an OWL class that is defined by OWL property constraints, which includes: i) the **hasID** property: it identifies an instance of the DObject. The hasID property is an OWL object property, whose range is *UniID* class of the ActOn ontology; ii) the **value** property: it contains a value of the corresponding dynamic property at a specified time period (i.e., lifetime). It is a datatype property, and its range is xsd:int (XML Schema datatype) [10, 3]; iii) the **timestamp** property: it is used to label the current time of the value of a dynamic property assigned, which in turn can be used to calculate the lifetime of the dynamic property. It is also a datatype property. The range of the property is xsd:time; The iv) the **queryType** property: it describes the embedded query that can be used to get an instance of a suitable information source in a ISO. The queryType property is an object property and its range is iso:InfoSrc; finally v) the **condition** property: it gives restrictions to the query, which allows to specify the query parameters. It is a datatype property and the range is xsd:string.

$$\begin{aligned} \textbf{DObject} \; & \exists \text{ hasID UniID} \\ & \exists \text{ value xsd:int} \\ & \exists \text{ timestamp xsd:time} \\ & \exists \text{ queryType iso:InfoSrc} \\ & \exists \text{ acton:condition xsd:string} \end{aligned} \tag{1}$$

The *UniID* class is used to identify an instance of DObject. It is a pair (EntityID, PropertyID), where EntityID to identify a class in a (DO), for example, *ComputerElement* in the EGEE domain ontology. The propertyID indicates an ActOn dynamic property, such as **freeCPU** in the example. The formalism definition of *UniID* in Description Logic is as follows:

$$\textbf{UniID} \; \exists \text{ hasID } (\text{EntityID} \sqcup \text{ProeprtyID})$$

Furthermore, three ActOn properties are defined in ActOn for DObject description, which are:

- The queryType property is an OWL Object property which points to the Class(InfoSource) in ISO. Its syntax in BNF can be:

```
queryType ::= 'ObjectProperty('individualvaluePropertyID
              {'domain('DO ClassID' ')'}
              {'range(' ISO ClassID ')'} ')'
```

- The value property is an owl:DatatypeProperty; it is a relation between an instance of DObject class and XML Schema datatypes. Its syntax in BNF can be:

```
value ::= 'DatatypeProperty('\textbf{datavaluePropertyID
          {'domain('DO ClassID' ')'}
          {'range(' xsd:Int ')'} ')'
```

- The timestamp property is an owl:DatatypeProperty, whose range is restricted to xsd:time. Its syntax in BNF can be:

```
timestamp ::= 'DatatypeProperty('\textbf{datavaluePropertyID
          {'domain('DO ClassID' ')'}
          {'range(' xsd:Time ')'} ')'
```

The above properties are mainly used to describe the dynamic relationships between the DO classes in ActOn. In the next section, we will show how a query can be generated from an instance of the *DObject* class with the properties in ActOn.

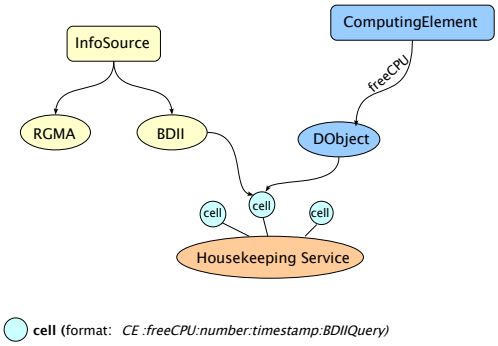

Figure 2. The ActOn Housekeeping service

In addition, as shown in Figure 2, the instances of the DObject are actually managed by a housekeeping service [12]. The housekeeping service creates/deletes/manages *cells* in order to manage instances of the DObject [6,

9]. Each *cell* is actually an RDF metadata that contains the UniID of the instance and its lifetime. In more detail, the housekeeping service will create a cell for an instance of the DObject. The cell keeps the instance of the DObject alive by managing its lifetime. Besides, a cell maintains a link between an instance of the DObject and an instance of *InfoSrc*, so that the update of the dynamic property can be executed automatically during the lifetime of a DObject instance.

4. Dynamic Management in ActOn by Examples

In this section, we give an example to illustrate how the DO and ISO can be linked in ActOn, and how the combination can support dynamic changes in a large-scale distributed system. In particular, we show how an embedded query for a particular instance of the ISO can be generated and queried. The example is a *ComputerElement* class, which we take from EGEE domain ontology. The class *ComputerElement* is defined by four property constraints, which can be described in Description Logic as follows:

$$\begin{aligned} \textbf{ComputerElement} \; & \exists \text{ supportVO VO} \\ & \exists \text{ requiredService SecurityInfras} \\ & \exists \text{ requiredService (JobMgt} \sqcup \text{Scheduler)} \\ & \exists \textbf{ freeCPU acton:DObject} \end{aligned} \tag{3}$$

So, we can describe the class *ComputerElement* in OWL as follows:

```
    <rdfs:subClassOf>
      <owl:Restriction>
          <owl:onProperty rdf:resource="#freeCPU"/>
          <owl:someValuesFrom rdf:resource="#DObject"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="&xsd;string"
      >computing element</rdfs:comment>

<owl:Class rdf:ID="DObject">
    <rdfs:subClassOf rdf:resource="#ActOn"/>
</owl:Class>
```

By the definition of *ComputerElement*, the DObject can be in turn specified as follows:

$$\begin{aligned} \forall \; \textbf{ComputerElement} \sqcap \exists \text{ middleware.gLite} \longrightarrow \\ \textbf{DObject} \; \exists \text{ hasID UniID} \\ \exists \text{ value xsd:int} \\ \exists \text{ timestamp xsd:time} \\ \exists \text{ queryType infoSrc:BDII} \\ \exists \text{ acton:condition “belongSite”} \end{aligned}$$

where: the format of UniID of this DObject in BNF is as follows:

```
UniID::=(CEID:freeCPUID);
```

So the target information server of this property freeCPU is a BDII server; the query condition for answering an instance of the BDII server from information source ontology (i.e., ISO) is the distance (e.g., property belongSite) between the instance of the *ComputerElement* class and the instance of the *BDII* class. More precisely, it will search for an instance of the *BDII* class within the same site as the instance of *ComputerElement* class.

Now we turn to the ISO part. We have a class *BDII*, which is a subclass of the *InfoSource* class. It is the class for BDII services in the EGEE infrastructure. BDII is a information service of gLite middleware, which is used in the EGEE Grid infrastructure to provide information mainly about Grid computing resources and storage resources. The definition of the *BDII* class in description logic is presented as follows:

BDII $\exists$ accessPoint URI
$\exists$ accessAPI xsd:String
$\exists$ acton:aboutEntity egee:ComputerElement
$\exists$ acton:aboutProperty egee:freeCPU
$\exists$ egee:belongSite xsd:string

As defined in the *ComputerElement* class, it has the property freeCPU, which is associated with an ActOn DObject, as shown in Figure 1. So the values of freeCPU property is a tuple {uniID, value, timestamp, infoSrc, condition}. Following this definition, we can generate instances of *ComputerElement* class. The example we take is from the EGEE CY-01-KIMON site. In the CY-01-KIMON Grid node of the EGEE Grid infrastructure [8], we have a computer element named ce101.grid.ucy.ac.cy. We describe the ce101.grid.ucy.ac.cy instance based on the definition of the *ComputerElement* class. It supports VOs and has several required services running on it. It has time a sensitive property which is assigned to a DObject to the property. Besides giving a value of free CPU number to freeCPU, we also give information about the lifetime of the property, the information server may be and its query condition, and a uniform ID to identify it. We can create an instance of the class ComputerElement described in OWL as follows:

```
<owl:Class rdf:ID="BDII">
    <rdfs:subClassOf rdf:resource="#InfoSrc"/>
</owl:Class>
<BDII rdf:ID="bdii101.grid.ucy.ac.cy">
    <belongSite xml:lang="en">EGEE.UCY</belongSite>
    <accessPoint xml:lang="en"
        >bdii101.grid.ucy.ac.cy:2170</accessPoint>
    <accessProtocol xml:lang="en">ldap</accessProtocol>
    <schema xml:lang="en">glueSchema</schema>
    <dataModel xml:lang="en">ldap</dataModel>
</BDII>

<owl:Class rdf:ID="ComputerElement">
    <rdfs:subClassOf rdf:resource="#GridComponent"/>
    <rdfs:subClassOf>
```

```
        <owl:Restriction>
                <owl:onProperty rdf:resource="#freeCPU"/>
                <owl:someValuesFrom rdf:resource="#DObject"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
    <ComputerElement rdf:ID="ComputerElement_ce101_ucy">
        <freeCPU rdf:resource="#DObject_freeCPU_ce101_ucy"/>
    </ComputerElement>
    <owl:DatatypeProperty rdf:ID="dataModel">
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:ID="DObject">
        <rdfs:subClassOf rdf:resource="#ActOn"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#lifetime"/>
                <owl:maxCardinality rdf:datatype="&xsd;int">1</owl:maxCardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
    <DObject rdf:ID="DObject_freeCPU_ce101_ucy">
        <lifetime rdf:datatype="&xsd;dateTime"
            >2007-10-02T12:10:08</lifetime>
        <value rdf:datatype="&xsd;int">123</value>
        <belongSite xml:lang="en">EGEE.UCY</belongSite>
        <hasID rdf:datatype="&xsd;string"
            >ce101:freeCPU:121008</hasID>
        <queryType rdf:resource="#bdii101.grid.ucy.ac.cy"/>
    </DObject>
```

From the above description, we can have the exact values of the DObject for the instance (ce101.grid.ucy.ac.cy) of *ComputerElement* class. The UniID of this DObject is ce101.grid.ucy.ac.cy, freeCPU, bdii101.grid.ucy.ac.cy, CY-01-KIMON, where

- CEID is ce101.grid.ucy.ac.cy
- freeCPUID is rdf:ID="freeCPU"
- QueryBDII is an BDII server that is bdii101.grid.ucy.ac.cy
- the site name is “CY-01-KIMON”

Also, the value of the available CPU number is 123, and its time stamp is 2007-10-02T12:10:08.

The above information can then be used to generate a LDAP based search query as follows:

```
ldapsearch ldap://bdii101.grid.ucy.ac.cy:2170 -b mds-vo-name=CY-01-KIMON,o=grid
 'GlueCEName=ce101.grid.ucy.ac.cy' GlueCEStateFreeCPUs
```

The query can be executed by a BDII wrapper to fetch the value of the number of free CPU of ce101.grid.ucy.ac.cy of CY-01-KIMON site in EGEE Grid [12].

5. Conclusions and Future work

In this paper, we introduce the ActOn semantic model and illustrate how the model can be used to deal with the dynamicity in a large-scale distributed

system by examples from EGEE Grid. The ActOn semantic model can be suitable for representing metadata about entities and resources in a very dynamic distributed environment. To address dynamicity, ActOn embeds a query into the semantic metadata description. This allows the dynamic information to be updated dynamically according to its lifetime and update policy.

Our future work will focus on the lifetime estimation algorithm. In a very dynamic distributed system, lifetime of distributed resources and entities is not always fixed or known in advance. So a proper lifetime estimation can be a key issue to manage dynamic. We intend to design a lifetime adjustment algorithm to cope with the Grid environment.

Acknowledgements

This work is supported by the EU FP6 CoreGrid Network of Excellence (FP6-004265). We also thank other members of the IMG group at Manchester for their helpful discussions: Paolo Missier, Matthew Horridge, Bijan Parsia.

References

[1] OntoGrid Project. http://www.ontogrid.eu/.

[2] OntoGrid CVS. http://www.ontogrid.eu/ontogrid/downloads.jsp.

[3] Paul V. Biron and Kaiser Permanente and Ashok Malhotra (editors). *XML Schema Part 2: Datatypes - W3C Recommendation*. W3C Recommendation, October 2004.

[4] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. *OWL Web Ontology Language Reference*. W3C Recommendation, February 2004.

[5] O. Corcho, P. Alper, P. Missier, S. Bechhofer, and C. Goble. Metadata management: requirements and architecture. In *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing (Grid 2007)*, pages 97–104.

[6] O. Corcho, P. Alper, I. Kotsiopoulos, P. Missier, S. Bechhofer, and C. Goble. An Overview of S-OGSA: A Reference Semantic Grid Architecture. *Journal of Web Semantics*, 4(2):102–115, 2006.

[7] M. D. Dikaiakos, R. Sakellariou and Y. Ioannidis. Information Services for Large-scale Grids: A Case for a Grid Search Engines. In *Engineering the Grid: status and perspectives*, pages 571–585, American Scientific Publishers, 2006.

[8] Enabling Grids for E-sciencE (EGEE). http://public.eu-egee.org/.

[9] P. Missier, P. Alper, O. Corcho, I. Dunlop, and C. Goble. Requirements and services for metadata management. *IEEE Internet Computing*, 2007.

[10] P.F. Patel-Schneider, P. Hayes, and I. Horrocks. *OWL Web Ontology Language Semantics and Abstract Syntax*. World Wide Web Consortium, February 2004.

[11] P.F. Patel-Schneider and I. Horrocks. *OWL 1.1 Web Ontology Language Overview*. World Wide Web Consortium, May 2007.

[12] W. Xing, O. Corcho, C. Goble, and M. D. Dikaiakos. Acton: A semantic information service for EGEE. In *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing (Grid 2007)*, pages 81–88.

[13] W. Xing, O. Corcho, C. Goble, and M. D. Dikaiakos. Information quality evaluation for grid information services. In *CoreGRID Symposium 2007*. In conjunction with EuroPar 2007, pages 165–174.

[14] W. Xing, M. D. Dikaiakos, and R. Sakellariou. A Core Grid Ontology for the Semantic Grid. In *Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2006)*, pages 178–184, Singapore, May 2006. IEEE Computer Society.

[15] The Protégé Ontology Editor and Knowledge Acquisition System http://protege.stanford.edu/

[16] The Active XML Team. Active XML Primer. http://activexml.net/