ARCHITECTURAL PATTERNS FOR THE SEMANTIC GRID *

Ioannis Kotsiopoulos, Paolo Missier, Pinar Alper, Oscar Corcho,
Sean Bechhofer, and Carole Goble
School of Computer Science
The University of Manchester
United Kingdom
{ioannis, pmissier, penpecip, ocorcho, seanb, carole}@cs.man.ac.uk

Abstract The Semantic Grid reference architecture, S-OGSA, includes *semantic provisioning services* that are able to produce semantic annotations of Grid resources, and *semantically aware Grid services* that are able to exploit those annotations in various ways. In this paper we describe the dynamic aspects of S-OGSA by presenting the typical patterns of interaction among these services. A use case for a Grid meta-scheduling service is used to illustrate how the patterns are applied in practice.

Keywords: Semantic Grid, Grid services, architectural patterns.

*

1. Introduction

The Grid aims to support secure, flexible and coordinated resource sharing by providing a middleware platform for advanced distributed computing [6] . Grid middleware architectures aim to allow collections of any kind of resourcescomputing, storage, data sets, digital libraries, scientific instruments, people, etc to easily form Virtual Organizations (VOs) that cross organizational boundaries in order to work together to solve a problem. However, existing Grid middleware architectures and the standards on which they are based on, fall short of addressing some of the original vision of configurable, self-healing, adaptive, and interoperable middleware [6]. This is due mainly to the following reasons:

Knowledge burial. Knowledge and metadata regarding Grid entities is currently generated and used in an ad hoc fashion, much of it buried in the middleware's code libraries and database schemas. This esoteric expression and use of knowledge hinders interoperability when it comes to building open, interoperable and adaptive systems. Existing Grid middleware is therefore considerably affected by syntactic changes in protocols and representations, and it becomes highly dependent on human intervention during its operation.

Dominance of XML-based vocabularies and protocols. The Grid community has developed a number of specifications and standards that aim to increase interoperability among middleware components. XML has become the de-facto language not only for expressing these specifications, but also for describing Grid entities and their behaviour. However, XML-based specifications do not provide a complete solution to the problem of knowledge burial due to the lack of a shared formal interpretation of XML documents.

Lack of models for Grid processes. Many aspects of the Grid are still not formally defined, therefore it becomes difficult to identify the challenges and even more difficult to find solutions. Take as an example the formation of Virtual Organizations (VOs); creating a model for forming VOs can help setting-up a community-wide terminology, highlight differences among existing systems and bring about previously unforeseen issues to be solved for interoperability. This model should be the product of a knowledge acquisition process, similar to those being undertaken by the Web [4], Web Services [3]and Semantic Web Services communities [9, 14]. The outcome of the modeling process can be used for the development of interoperable metadata based on explicit semantics.

The Semantic Grid is an extension of the current Grid in which information and services are given well defined and explicitly represented meaning, better enabling computers and people to work in cooperation [8]. In the Semantic Grid, the goal of sharing virtualized computational and data resources is extended to include explicit metadata and knowledge. During the last few years, several projects have embraced this vision and there are already successful pioneering applications that combine the strengths of the Grid and of semantic technologies [15]. As a result of some of these efforts, the S-OGSA reference architecture has been recently proposed [5], with the aim of providing a systematic approach for designing Semantic Grid applications.

This paper is focused on the dynamic aspects of semantic Grid. We begin by presenting a summary of S-OGSA ("semantically enhanced OGSA"); then introduce a use case for Semantic Grid, namely semantic meta-scheduling of Grid resources [11]. With the help of the use case, we present two service interaction patterns that demonstrate the key aspects of Semantic Grid dynamics in S-OGSA. Finally, we provide some conclusions and future research directions.

2. Semantic Grid concepts

In this section we provide a summary of the fundamental properties of S-OGSA; a more comprehensive discussion can be found in [5]. S-OGSA consists of (i) an information model of semantic resources, which extends the OGSA model, and (ii) two new types of Grid services, *Semantic Provisioning Services* and *Semantically Aware Grid Services*.

2.1 A Semantic Grid Information Model

Two types of entities are at the basis of the information model:

Grid Entities (*G-Entities*) are anything that carries an identity on the Grid, including resources and services [19];

Knowledge Entities are special types of Grid Entities that represent or could operate with some form of knowledge. Examples of Knowledge Entities are ontologies, rules, knowledge bases or even free text descriptions that encapsulate knowledge that can be shared. Knowledge services are those that provide access to or operate over those knowledge resources, e.g. rule engines and automated reasoners.

Semantic Bindings (*S-Bindings*) are the entities that come into existence to represent the association of a Grid Entity with one or more Knowledge Entities. The existence of such an association transforms the subject Grid entity into a **Semantic Grid Entity**. Semantic Bindings represent metadata

assertions on web resources. In our model, Semantic Bindings are first class citizens as they are modelled as Grid resources with an identity and manageability features as well as their own metadata.

Semantic Grid Entities are those Grid Entities that are either the subject of a semantic binding, are themselves a semantic binding, or a Knowledge Entity. In keeping with our design principles, Grid entities can simultaneously be associated with zero or multiple knowledge entities of different forms and capabilities, and can acquire and discard associations with knowledge entities through their lifetime. It should be noted that S-OGSA does not prescribe any specific technology for the realisation of these.

2.2 Semantic Provisioning Services

These are services that provision semantic entities. These Semantic Services are themselves Grid Services. Following the aforementioned classification of semantic entities, two major classes of services are:
Knowledge provisioning services (*KPS*), which can produce (and in some cases store) knowledge resources, and that can be used to manage knowledge resources. KPS support the creation, storage and access of different forms of knowledge resources. For example: ontology services (a major form of knowledge) and reasoning services.

Semantic Binding provisioning services, which can produce (and in some cases store) S-Binding resources, and that can be used to manage S-Binding resources. For example: semantic binding index services, for accessing and storing metadata associating Grid entities with knowledge entities; and annotation services for generating metadata from different types of information sources, like databases, files or provenance logs. S-Bindings are stateful, so they are subject to soft state processes; i.e. they will time out, get deleted or be removed. A typical way of producing S-Bindings is by annotating Grid entities as is shown in the Grid entities annotation pattern (Section 4).

2.3 Semantically Aware Grid Services

This class of Grid Services are able to exploit semantic technologies to consume semantic bindings in order to deliver their functionality. Their role is complementary to the role of *Semantic Provisioning Services* since they consume the semantic entities held by **Knowledge provisioning services** and **Semantic Binding provisioning services** and use their services. The combination of *Semantic Provisioning Services* and *Semantically Aware Grid Services* can address the knowledge burial problem discussed in Section 1 since explicitly shared knowledge can be consumed by third party services. Semantically

Aware Grid Services are able to exploit explicit semantics, and therefore can benefit from the additional context it provides for service operation. Examples include:

- A **VO Manager service** that can perform semantics-aware service access authorization;
- A **Grid resource catalogue** that supports semantic searches;
- An **ontology service** that is capable of incorporating new concepts into an ontology.

3. The Grid scheduling use case

We illustrate the use of semantic grid concepts in practice, by describing an existing Grid service that is currently being enhanced as a semantics-aware service. The service addresses a real and common problem in the area of resource co-allocation on the Grid. The problem of resource co-allocation emerges when dealing with complex workflows that require multiple data, computing and network resources; these resources are commonly highly distributed, and are subject to autonomous and independent management by different organizations.

We are specifically interested in resources whose usage is controlled by schedulers on the Grid, either at the local or the cluster level; allocating multiple such resources and orchestrating their access requires the introduction of a new type of Grid service, called a *meta-scheduler* (MS) or *super-scheduler*. The MS is responsible for the co-scheduling [17] of resources in order to assemble, on demand, a virtual machine that enables the execution of distributed jobs consisting of many parallel tasks. In particular, the MS provides higher-level resource management by implementing a consistent interface into various Grid scheduling systems, and thus hides much of the heterogeneity of the local schedulers that control the actual underlying resources.

For our use case, we focus on the generic meta-scheduler recently proposed by Wäldrich et al [20], whose design attempts to generalize on the type of resources that can be scheduled. This MS interfaces with multiple local schedulers, negotiating with them *advance reservation* of resources based on user requirements that may include time and QoS constraints. The goal of the negotiation is to determine time slots where the required resources are available for the requested start times of the application or workflow parts. The meta-scheduler implements two main functionalities: (i) allocation of a single resource for a single application for a fixed period of time, and (ii) co-allocation of multiple resources for the same fixed period of time for single or multiple applications.

In order to be able to participate in the negotiation, schedulers must satisfy at least the first of the following requirements:

1 provide advance reservation of resources by offering job execution start and stop times;

2 allow at least partial access to the local schedules, e.g. the available timeslots;

3 allow for some control on existing reservations, e.g. by handling requests for cancellation, or time extensions.

Thus, meta-scheduling includes the following main steps:

- discover schedulers that (i) manage resources that are compatible with the requirements of the Grid workflow, and (ii) satisfy (at least) the first of the remaining two requirements above;
- negotiate suitable timeslots with the pre-selected schedulers;
- commit to the advance reservation, and interact with the schedulers to handle any subsequent change in the agreed-upon reservation.

The meta-scheduler interacts with local schedulers through dedicated adapters that hide the heterogeneity of the schedulers' native interfaces. These adapters offer a uniform set of abstract operations to the meta-scheduler, which include requesting available start time slots for jobs, submitting scheduling requests for a specific time slot, and requesting the state of the current reservation.

The meta-scheduler described in [20] negotiates with the local adapters using the WS-Agreement framework [1]. It has been integrated into the UNICORE Grid system, and its functionality has been demonstrated on the VIOLA testbed for advanced network services [12]. The meta-scheduler is accessible through UNICORE client plugins, which allow users to specify requests for co-allocated resources to run a distributed job on VIOLA.

3.1 Limitations of the current meta-scheduling model

The focus of the current implementation is on the meta-scheduling algorithm, rather than on the discovery and pre-selection of the eligible schedulers, and on the design of the adapters. However, the latter is a serious issue for the scalability of the proposed approach. Our study of meta-scheduling as a promising Semantic Grid use case stems from the observation that, while the adapters provide a uniform set of operations, no shared data model is available to describe a scheduler's set of capabilities. For example, there is no explicit and shared definition of scheduling concepts like *timeslot* or *schedule queue*,

or of capabilities like *timeslot reservation change*. Instead, these concepts are left implicit in the implementation of the adapters, which only expose a simple set of scheduling operations.

This arrangement results in an architecture that is vulnerable to changes. Firstly, when the schedulers' capabilities change, they are not easily reflected in the adapters, which leave this knowledge implicit within their code. Secondly, when the meta-scheduler requirements for the required capabilities change, eg due to changes in the meta-scheduling negotiation strategy or allocation decision algorithm, there is no shared vocabulary to describe the new requirements.

Motivated by these observations, we have proposed [11] a semantic approach to meta-scheduling on the Grid, which improves upon the current design by:

- introducing a shared, explicit and lightweight but extensible semantic model to describe a scheduler's set of capabilities as well as its current state (which the meta-scheduler will need to query, see requirement 2). This is known as the *Grid Scheduling Ontology*;
- enhancing the adapters so that they can generate metadata regarding the schedulers' capabilities;
- enhancing the existing meta-scheduler as a Semantic Grid service, which is (i) aware of the available schedulers' semantics annotations, and (ii) able to exploit them to perform scheduler discovery and pre-selection.

3.2 The Grid Scheduling semantic model

The design of the enhanced meta-scheduler is based on a semantic model of Grid scheduling concepts. A detailed presentation of the model can be found in [11]; what follows is a brief summary.

At the core, the model includes concepts for scheduler, scheduler capabilities, scheduler reservation, and additional concepts to represent the state of a local schedule; each of these classes is the root of an extensible hierarchy. Furthermore, relationships amongst these root classes are established using *object properties*, used for instance to associate sets of capabilities to a scheduler.

The model is defined as an ontology in OWL DL [12]; using the OWL DL operators, scheduler classes can be defined to contain all and only schedulers with a defined set of capabilities. For example, a limited-disclosure-scheduler, a subclass of scheduler, is the class of all schedulers that allow their local schedule to be queried.

These intensional definitions provide a focused way to add *semantic annotations* to individual schedulers, which are instances of one or more of the scheduler classes. In their simplest form, annotations include capabilities metadata,

which may state for instance that that “a scheduler is both capable of offering advance reservation, and allows queries on its current schedule”.

These annotations facilitate the schedulers’ pre-selection by a meta-scheduler. More precisely, they allow a Description Logic reasoner [2] for the type of DL supported by OWL, to automatically classify a scheduler whose semantic annotations are known, as a member of one or more scheduler classes, defined intensionally as shown above. Once this classification has taken place, it is easy to show that scheduler discovery using this model amounts to (i) selecting from the ontology a scheduler class whose definition satisfies the selection criteria, and (ii) querying the ontology class to retrieve all the individual schedulers in the class.

Casting this discovery pattern within the Semantic Grid context is straightforward: local schedulers (LS) are Grid entities, and their semantic annotations can be defined as knowledge entities using the terminology introduced earlier; they are maintained in a metadata store as first-class Grid Entities themselves. Semantic bindings in this case embody the association between schedulers and their annotations; the bindings are exploited by the meta-scheduler, which becomes a semantically -aware and -capable Grid service.

Figure 3 shows how the meta-scheduler may make use of the S-OGSA semantic services suite presented in Section 4. In the next section, S-OGSA style interactions are described in a principled way, using the Grid meta-scheduling case study as an example.

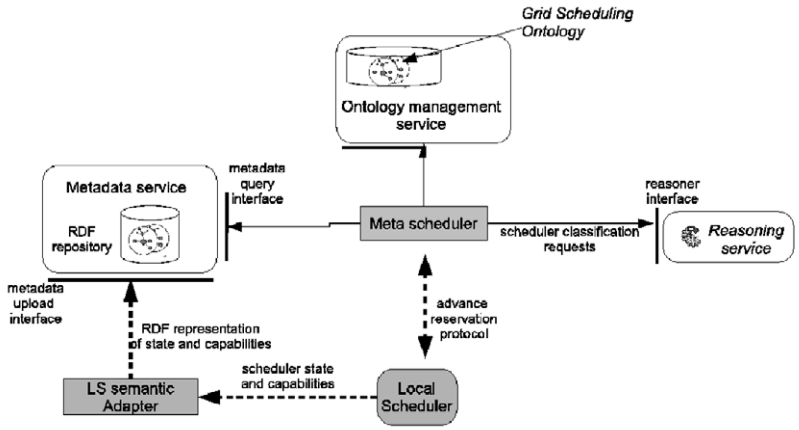

Figure 1. Casting meta-scheduling in the S-OGSA context

4. Service interaction patterns for the Semantic Grid

The description given in Section 2 provides a static view of the S-OGSA architecture. Ultimately, however, the goal of provisioning and consuming semantics in the Grid is realized when S-OGSA services interact with one another and with Grid entities. We now present the two most relevant service interaction patterns that define these dynamic aspects of the Semantic Grid. The patterns follow the main steps in semantic information processing in S-OGSA, namely:

- Producing semantic annotations, i.e., ontology-referenced metadata for some Grid entity (resources or services), and representing those annotations as persistent knowledge entities. Grid entities and their annotations are thus both first-class Grid citizens, and can participate in a semantic binding;
- Resolving the semantic bindings in order to retrieve annotations for given Grid entities.

These patterns describe the preparatory actions that any semantically-aware Grid service, such as the meta-scheduler and the adapters that are responsible for producing the metadata, would carry out before semantics can be exploited.

The patterns are presented according to the well-known format discussed in [16]. The dynamics of each pattern are explained with UML sequence diagrams while additional comments are used inside the sequence diagrams wherever the interaction is complex or needs some clarification, so as to make the diagrams as self-contained as possible.

4.1 Grid entity annotation pattern

Definition: The Grid entity annotation pattern encapsulates the functions needed to annotate Grid data resources or services, producing either raw or semantic metadata and store them persistently. By *raw metadata* we mean any annotation that can be associated to a piece of data, or, more generally, to a Grid entity. *Semantic metadata*, on the other hand, is metadata that carries explicit references to the semantic models, i.e., reference ontologies, required for its interpretation. In this work, we are only interested in the latter. When annotations are stored in a Metadata store they become Grid Resources since they are given a unique identifier. From this set of annotations, those that link Grid Entities with Knowledge Entities are called Semantic Bindings.

Example: The capability profile of a scheduler can be expressed using terminology from a Grid Scheduling Ontology (GSO), so that any user who has access to the ontology may be able to interpret the profile. The LS (Local Scheduler) Semantic Adapter shown in Figure 3 supervises and monitors one

local scheduler and produces semantic annotations regarding its capabilities and state changes. Annotations are used by the metascheduler for service pre-selection.

Context: Generation and storage of semantic metadata for Grid entities.

Problem: The use of intelligent reasoning mechanisms requires semantic metadata.

Solution: The annotation process can be either done manually, semi-automatically or on-demand without any user interaction. This pattern is concerned with automatic and semi-automatic annotations according to which an Annotation Service is able to fetch reference ontologies from an Ontology Service, and use them to create semantic annotations that can be interpreted using those ontologies. The outcome of the annotation is persistently stored using the Metadata Service.

Dynamics: The annotation process is triggered by a requestor that wants to annotate a piece of data. First, the annotation service needs to obtain a reference to a suitable ontology. For this, it invokes the Ontology Service which returns a handler to this ontology. During the annotation process this handler is used to retrieve ontology *concepts* and *properties* from the Ontology Service. Optionally, the Annotation Service may also retrieve existing annotations from the Metadata Service, for reference or for updating purposes. When the annotation process finishes, the annotation is persistently stored in the Metadata Service and assigned a unique identifier. The annotation has now become a special type of Grid Entity that links Knowledge Entities (i.e. the ontology) to Grid Resources. This Grid Entity is called a *Semantic Binding* and can be retrieved from the Metadata Service using the aforementioned unique identifier: the Semantic Binding ID.

Associated to the annotation is also, potentially, its provenance metadata, describing the annotation process itself (when it was performed, by whom, the external resources it is based on, and so forth). Note that this pattern is only concerned with S-OGSA service interactions, rather than with the specific annotation process, which may vary depending on the domain of the Grid resource and the purpose of the annotation.

4.2 Metadata and Knowledge querying Pattern

Definition: This pattern allows an application to retrieve semantic bindings and/or query the semantic metadata associated to a set of Grid Entities, with

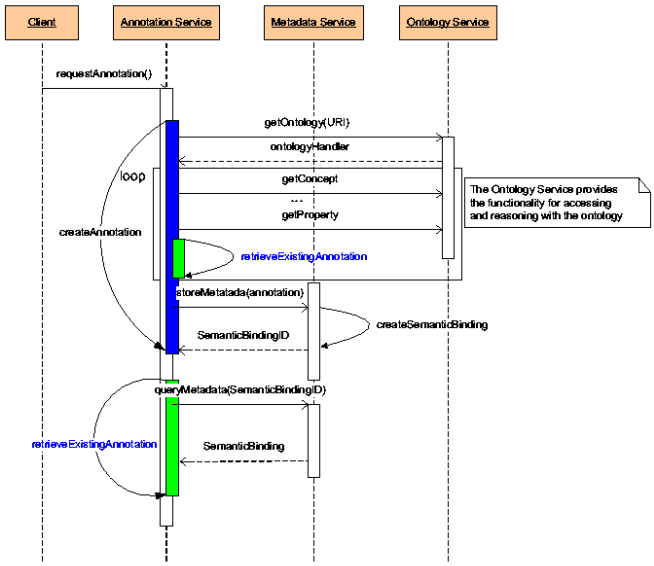

Figure 2. Data annotation interaction patterns

the help of the metadata and ontology services.

Example: The capabilities required for a scheduler to participate in advance reservation are represented by one or more scheduler classes in the ontology. As we have shown, eligible schedulers are all and only the instances of those classes. Retrieving those instances may require reasoning capabilities, as well as access to the metadata storage.

Context: Semantically aware Grid services that need to retrieve semantic bindings in order to perform their function.

Problem: Exploiting semantic bindings involves retrieving semantic metadata associated to some Grid entity.

Solution: Since semantic metadata can be implemented in a formal language (e.g., RDF Schema, OWL), reasoning techniques can be used in order to retrieve the metadata. Depending on the reasoning mechanisms available for the formal language in which the metadata is implemented, different types of inferences will be available, from the retrieval of subclasses or ancestors

of a given class to the classification of sets of individuals according to their most specific class. During the query answering process, we can exploit the reasoners capabilities in order to infer new facts by aggregating knowledge already stored in the Metadata Service.

Dynamics: The behaviour of the Metadata Querying Pattern is shown in Figure 3. Retrieving raw metadata is straightforward. For semantic metadata, the metadata service uses the ontology service for expanding or restricting the queries that are sent in the message, such as adding subclasses of the concepts used in the query, or detecting inconsistencies in the query before they are issued to the metadata service.

5. Discussion

In this paper we have provided a dynamic view of the Semantic Grid by focusing on some of the most common interaction patterns among the semantic middleware components identified in S-OGSA. Our coverage of patterns here is far from being exhaustive and there are several variations to undertaking the two core Semantic Grid functionalities covered in this paper, namely annotation and metadata querying.

The annotation (or metadata generation) pattern that we have covered displays the case, where metadata for grid entities is generated semi-automatically and on-demand, which in the illustrative scenario corresponds to the LS Semantic Adapter's annotation of a scheduler that has recently joined the VO, or to a scheduler that has just changed its state. On-demand and semi-automatic characteristics require the metadata generation pattern to include phases for discovery of annotation resources (e.g. ontologies).

Annotation could also be done automatically and initiated dynamically as Grid entities come into existence. Cases where VO membership of Grid entities change frequently; where most middleware activities heavily rely on existence of grid entity metadata, or where metadata represents historical/contextual information of a Grid entity (e.g. provenance), all necessitate annotation to be a sustained activity. In the sustained annotation case, the resource discovery phase is generally skipped, and the annotation tooling is configured to use a specific set of resources and methods.

The metadata querying patterns we have covered demonstrated capabilities ranging from simple retrieval of raw metadata to expansion of semantic metadata via ontological inference.

The patterns are intended to be the building blocks of more complex interactions that build-up **activities** of middleware and applications in the Semantic Grid ecosystem. For instance, in our illustrative scenario, the motivation for

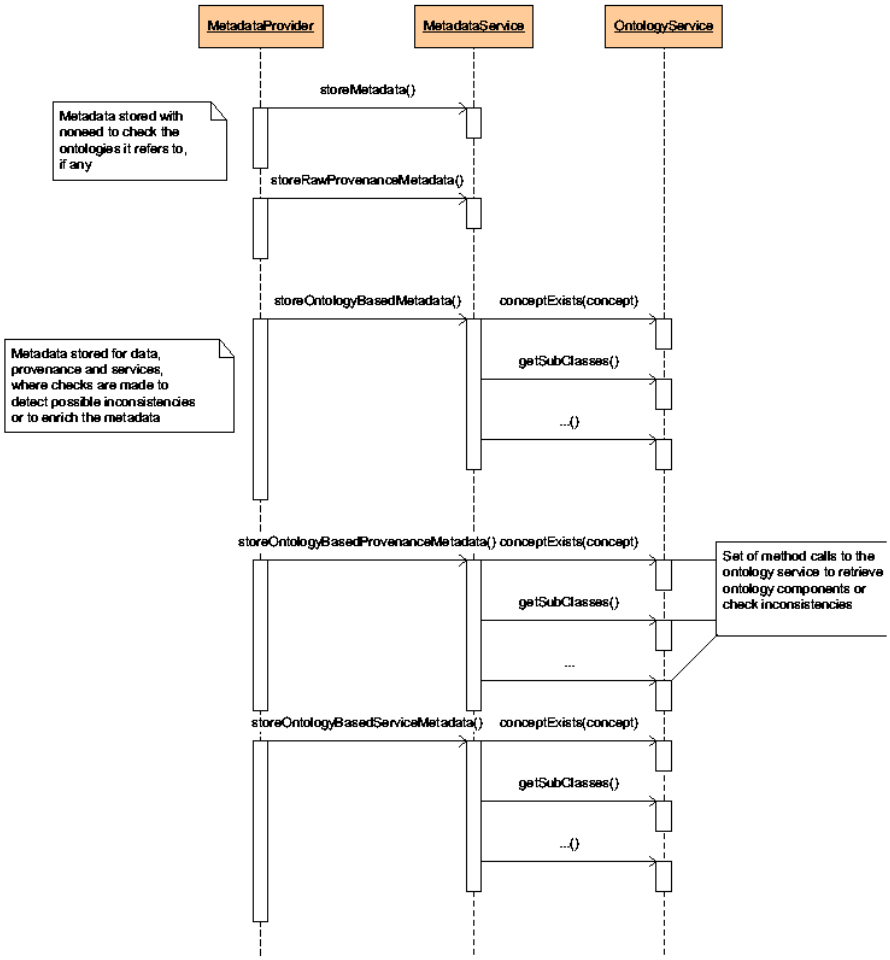

Figure 3. Metadata storage interaction patterns

providing semantic descriptions for local schedulers came from the need for the Semantic Discovery activity.

Such semantically-enhanced activities are also currently being investigated in fields such as the Semantic Web (SW) and Semantic Web Services (SWS). Work in these areas investigates *i)* suitable technologies and models for semantically describing resources in their respective distributed environment (e.g., the Web or the Web services) and *ii)* how these semantic descriptions can be exploited in the context of a particular activity with special focus

on discovery, negotiation and composition. There are certain aspects of the Grid that appear to have higher priority when compared to other distributed environments. These aspects and their effect on semantics can be summarized as follows:

Dynamism and Dependency Management. Unlike other distributed environments, the resources in the Grid are very dynamic. Resource state changes frequently, and information regarding the state of the system has a definite lifetime. Grid information systems aggregate resource state information (generally represented as XML based resource properties) into index services in order to provide an aggregate system snapshot and enable discovery of resources based on their properties. To the extent that semantic metadata adds to resource state information, managing the lifetime of the semantic bindings becomes important. These issues, which have so far attracted little research attention in the semantic web (SW) and semantic web services (SWS) communities, need to be addressed in the Grid context.

Trust and Consistency. Building a well-controlled resource sharing environment is the main aim in the Grid. Introduction of metadata and knowledge into the Grid brings about the issues of trust-ability and consistency of these. These issues are also under investigation in the SW and SWS communities [7]. The uptake of semantics in the Grid depends on existence of usable models and frameworks in this area.

6. Future Directions

Our current S-OGSA architectural descriptions, including their static and dynamic aspects, do not prescribe any semantic technology and content for the realization of semantic entities and services in the Grid. We are aware that the guidance of S-OGSA would increase if it is accompanied with some generic content and experience reports (e.g. best practices) on particular technology choices. Therefore, as part of our future work we will be providing:

Meta-models for knowledge and metadata. In order to facilitate interoperable use of the S-OGSA entities in a Grid environment we need to provide minimal information on what they are. This will be done by modelling the different types of realizations for Semantic Bindings (e.g., RDF, natural language) and Knowledge Entities (e.g., ontologies, rule bases).

Profiles for S-OGSA. In this chapter we have demonstrated S-OGSA with a scenario where Description Logic based knowledge and RDF based metadata representations have been used to provide semantic capability descriptions for

schedulers and their discovery through use of a DL classifier. The choice of realization technology for knowledge and metadata modelling depends on many factors including the nature of the problem at hand, the characteristics of the candidate semantic technologies and the availability and maturity of their associated tools/services. Returning to our example in this chapter, the use of an open-world based DL classifier proved suitable for discovery of distributed resources. This however should not imply that these particular technologies are fit for the solution of other Grid problems (e.g. policy reconciliation, agreement negotiation). In fact closer investigation of such problems [7]has shown that semantic technologies other than DL and RDF could be ideal for tailoring solutions to these problems. Based on this observation we would like to provide profiles for S-OGSA that demonstrate exploitation of different semantic technologies for the solution of different Grid problems.

References

[1] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web services agreement specification. Technical report, Global Grid Forum, July 2005. https://forge.gridforum.org/projects/graap-wg/document/WS-AgreementSpecification/en/16.

[2] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[3] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. Web services architecture. Available at http://www.w3.org/TR/ws-arch/, 2004.

[4] D. Brickley and R. V. Guha. Rdf vocabulary description language 1.0: Rdf schema. Available at: http://www.w3.org/TR/rdf-schema/, 2004.

[5] O. Corcho, P. Alper, I. Kotsiopoulos, P. Missier, S. Bechhofer, D. Kuo, and C. Goble. An overview of s-ogsa: a reference semantic grid architecture. *Journal of Web Semantics*, 4, 2006.

[6] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. V. Reich. The open grid services architecture, version 1.0. Technical report, Open Grid Services Architecture WG, Global Grid Forum, 2005.

[7] R. Gavriloaie, W. Nejdl, D. Olmedilla, K.E. Seamons, and M. Winslett. No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. In *1st European Semantic Web Symposium (ESWS2004)*, pages 342–356. Springer-Verlag, 2004.

[8] C. A. Goble, D. D. Roure, N. R. Shadbolt, and A. A. Fernandes. *In The Grid 2: Blueprint for a New Computing Infrastructure Second Edition*, chapter Enhancing Services and Applications with Knowledge and Semantics. Morgan Kaufmann, i. foster and c. kesselman, edition, 2003.

[9] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara. Bringing semantics to web services: The owl-s approach. In *First International Workshop on Semantic*

Web Services and Web Process Composition (SWSWPC 2004), San Diego, California, USA, 2004.

[10] D. L. McGuinness and F. v. Harmelen. OWL Web Otology Language Overview, February 2004. W3C Recommendation.

[11] P. Missier, P. Wieder, and W. Ziegler. Semantic support for Meta-Scheduling in Grids. Submitted.

[12] Online. VIOLA - Vertically Integrated Optical Testbed for Large Application in DFN, 2005. Project web site: http://www.viola-testbed.de/.

[13] L. Pouchard, L. Cinquini, and G. Strand. The earth system grid discovery and semantic web technologies. In *Workshop for Semantic Web Technologies for Searching and Retrieving Scientific Data, at the 2nd International Semantic Web Conference*, 2003.

[14] D. Roman, U. Keller, H. Lausen, J. d. Bruijn, R. Lara, M. Stollberg, A.Polleres, C. Feier, C. Bussler, and D. Fensel. Web service modelling ontology. *Journal of Applied Ontology*, 1:77–106, 2006.

[15] D. De Roure, Y. Gil, and J. A. Hendler. Guest editors' introduction: E-science. *IEEE Intelligent Systems*, 19:24–25, 2004.

[16] D. C. Schmidt, M. Stal, H. R., and F. Buschmann. *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, volume 2. John Wiley and Sons Ltd, 1 edition, 2000.

[17] J. Schopf. Ten Actions When Grid Scheduling – The User as a Grid Scheduler. In J. Nabrzyski, J. Schopf, and J. Weglarz, editors, *Grid Resource Management – State of the Art and Future Trends*, pages 15–23. Kluwer Academic Publishers, 2004.

[18] N. Sharman, N. Alpdemir, J. Ferris, M. Greenwood, P. Li, and C. Wroe. The mygrid information model. In *UK e-Science All Hands Meeting*, 2004.

[19] J. Treadwell. Open grid services architecture glossary of terms. Technical report, Open Grid Services Architecture WG, Global Grid Forum, 2005. Available at: http://forge.gridforum.org/projects/ogsa-wg.

[20] O. Wäldrich, P. Wieder, and W. Ziegler. A meta-scheduling service for co-allocating arbitrary types of resources. In *Proc. of Sixth International Conference on Parallel Processing and Applied Mathematics (PPAM 2005)*, 2005.