

Building Large-scale, Service-Oriented Distributed Systems using Semantic Models

Carole Goble¹, Dean Kuo¹, Savas Parastatidis², Paul Watson²

Version 1.0

¹ School of Computer Science, University of Manchester

² School of Computing Science, University of Newcastle upon Tyne

Executive Summary

Service-oriented computing is a set of principles for building loosely-coupled distributed applications through the composition of autonomous services. For example, OGSA aims to define a core set of composable services for the Grid application domain that includes job submission and monitoring, accounting, reservation, execution planning and database access, and the goal is for application-developers to be able to extensively re-use these services when building large scale service-oriented distributed systems without duplicating existing functionality. Over time, there is the hope that a healthy ecosystem of Grid services will be developed, with rich new services feeding off the capabilities of existing services, and the new services themselves providing the basis for future higher-level services.

The tasks of discovering and composing services to meet a distributed application's functional and non-functional requirements are not always straightforward. This is especially true in a distributed environment where services are autonomous – that is, they have been designed, implemented, deployed and managed in different administrative domains – and is currently difficult as:

- there is a lack of a machine-processable description of a service's functional and non-functional characteristics;
- there are no set of simple, standardised and composable conceptual models that can succinctly describe a service's functional and non-functional characteristics.

Current distributed applications are often developed by combining services using a “trial and error” paradigm where the overall behaviour of the application is likely to be an emergent property and can't be predicted through knowledge of the services from which it is composed.

In this paper we argue that these problems should be addressed by providing semantic descriptions of services as metadata. Such metadata will capture a service's functional and non-functional characteristics, including the service's externally visible process model, data semantics, policies, service level agreements and quality of service guarantees. Associated tooling will be needed to assist architects and developers in creating (semi-automatically or automatically), publishing, updating, managing and storing such metadata as the service evolves through its lifecycle. Tooling will also be required to support application builders in discovering services that meet given criteria and in deriving the overall behaviour of a service-oriented application.

While there is existing work that assists in meeting this goal, we recommend further research and development in a number of areas including:

- Well-defined, simple, composable and unambiguous semantic models that describe a service's functional and non-functional properties.
- Development and runtime tooling to support metadata creation, publication and discovery, semantic composition and decomposition, and behaviour prediction.
- Tools and techniques to support run-time monitoring and evolution of applications, based on the semantic models.
- An investigation into existing best practice in service-based and Grid applications in order to understand how the proposed approaches should be utilised in order to improve design, implementation and deployment.

Table of Contents

1. Introduction.....	7
2. The Current State of OGSA.....	8
3. The Need for Semantic Models.....	9
4. Models	11
4.1. Legal Contracts	11
4.2. Service Contracts	12
4.3. Security	12
4.4. Service Information Model.....	13
4.5. Information Modelling.....	13
4.6. Summary	13
5. Metadata	14
5.1. State of the Art	14
5.2. Semantic Grid.....	15
5.2.1. Semantic Web.....	15
5.2.2. Semantically Aware Services and Protocols.....	16
5.2.3. Open Issues	16
6. Case Studies.....	17
6.1. GOLD.....	17
6.2. ^{my} Grid.....	17
6.3. RealityGrid	18
6.4. JISC eLearning Framework	19
7. Conclusions and Recommendations	19
Acknowledgements.....	19
References	20

1. Introduction

Over the last few years the Global Grid Forum (GGF) Open Grid Services Architecture (OGSA) working group has been documenting the “requirements, functionality, priorities, and interrelationships” [1] of the key services necessary to build Grid systems. The OGSA documents and roadmap guide the ongoing standardisation efforts within GGF for these key services. Because of the work that has been done within the GGF working groups, we now have a better understanding of the architectural approach and suite of underlying technologies required for building the Grid fabric and applications.

While the slow, careful, and important, standardisation work takes place for the basic Grid core services, this paper considers approaches and technologies that will simplify the construction of large-scale, service-oriented distributed systems for these and other services. The approaches and technologies are in line with the vision and future directions for the future of European Grids in evolving Grids towards a wider and more ambitious vision of Service Oriented Knowledge Utilities (SOKU) [2].

The key aim of service-oriented computing is to define a set of principles for the development of large-scale distributed applications through the composition of autonomous services. The goal is that application-builders will be able to extensively re-use services – building applications by combining existing services rather than by duplicating their functionality. Over time, there is the hope that a healthy ecosystem of grid services will be developed, with rich new services feeding off the capabilities of existing services, and the new services themselves providing the basis for future, higher-level services.

The tasks of discovering and composing services to meet the functional and non-functional requirements of a distributed application are not always straightforward. This is currently difficult as:

- there is a lack of a machine-processable description of a service’s functional and non-functional characteristics;
- there are no simple and standardised conceptual model that can describe, with clarity, a service’s functional and non-functional characteristics.

As a result, it is difficult to discover a service’s functional and non-functional characteristics when combining them into a new distributed application. This is especially true in a distributed environment where services are autonomous, that is they have been designed, implemented, deployed and managed in different administrative domains.

Current distributed applications are often developed by combining services using a “trial and error” approach where the overall behaviour of the application is likely to be an emergent property. The behaviour can’t be predicted through knowledge of the services from which it is composed.

In the remainder of this paper, we argue that these problems should be addressed by providing semantic descriptions of services as metadata. Such metadata will capture functional and non-functional properties of the service, including the service’s externally visible process model, data semantics, policies, service level agreements and quality of service guarantees. Associated tooling will be needed to assist architects and developers in creating (automatically or semi-automatically), publishing, updating, managing and storing such metadata as the service evolves through its lifecycle. Tooling will also be required to support application builders in discovering services that meet given criteria and in deriving the overall behaviour of a service-oriented application.

While there is existing work that assists in meeting these goals, we recommend further research and development in a number of areas including:

- well-defined simple, composable and unambiguous semantic models that describe a service’s functional and non-functional properties;

- development and runtime tooling to support metadata publication and discovery, semantic composition and decomposition, and behaviour prediction;
- tools and techniques to support run-time monitoring and evolution of applications, based on the semantic models;
- an investigation into existing best practice in service-based and Grid applications in order to understand how the proposed approaches should be utilised in order to improve design, implementation and deployment.

The rest of the paper is structured as follows. The architectural work of the OGSA working group is briefly presented in Section 2. Section 3 presents the requirements and benefits of using semantic models to describe and compose services. Section 4 then describes a set of models that address some generic aspects of service-oriented computing, specifically: legal contracts, service contracts, security and information. Metadata is at the core of declarative models, as is described in Section 5, which describes methodologies and infrastructure for supporting the exploitation of semantic models. The use and potential of the model-based approach with regard to three existing e-science applications is then described in Section 6. Finally, Section 7, concludes with a summary of the discussion and our recommendations.

2. The Current State of OGSA

The Open Grid Services Architecture (OGSA) represents a service-oriented platform, suite of services and approaches for building Grid applications. This includes describing how computational jobs are executed and managed, usage logged and charged to users, data accessed and managed, and security requirements are expressed through policies and enforced at runtime.

While the architecture is technology-agnostic, Web Services has been chosen as the infrastructure layer upon which the OGSA services are implemented. This is due to the promise of interoperability, significant industry investment, tooling, documentation and stability that are associated with these technologies.

As the GGF community is focussed on creating standards for high-level services, work on lower-level infrastructure specifications (e.g. SOAP [3], WSDL [4], WSRF [5], WS-Notification [6], WS-Transfer [7], WS-Eventing [8], WS-DistributedManagement [9] and WS-Security [10]) is left to standardisation bodies such as OASIS [11] and W3C [12]. Nevertheless, given that the work on these infrastructure specifications is not yet complete, the GGF community has to make decisions on which of them to adopt and how to use them, at least until the Web Services Interoperability organisation [13] produces universally accepted profiles for all these Web Services specifications. As a result, OGSA is working on profiles of Web Services specifications for building Grid services.

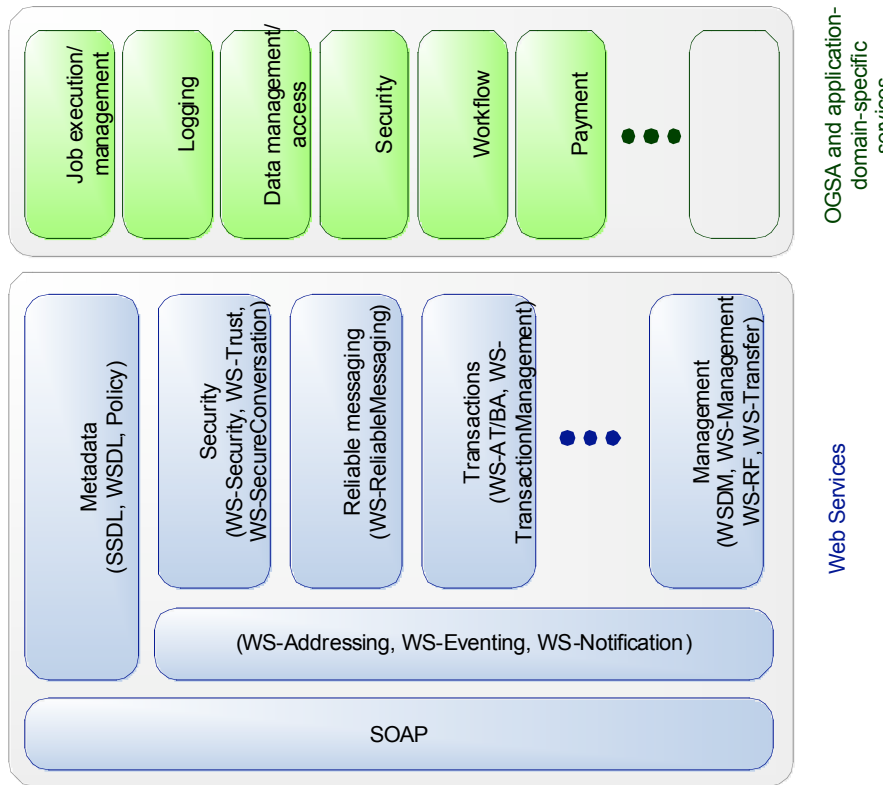


Figure 1: A layered view of the infrastructure and OGSA services specifications

3. The Need for Semantic Models

Outside of the GGF, we are currently witnessing a transition from the Web to “Web 2.0” and “Semantic Web”. One of the main principles of Web 2.0 is “the web as a platform of services” in which distributed applications are built by composing services. Rather than buy software licenses, users would instead pay for the usage of these services. In support of this vision, the Semantic Web community is developing standards, protocols and components for knowledge management where knowledge is encoded as machine-processable metadata. This has the aim of simplifying the process of finding and composing services.

There is great interest in applying the same ideas that drive the changes on the Web to the way e-Science and Grid distributed applications are built, deployed and managed over their complete lifecycle. Due to the focus on service and resource integration, service-oriented applications are continuously increasing in complexity. With a requirement to dynamically evolve in an ever changing runtime environment, there is the need for more abstract and semantically-rich ways to perform system design, development, deployment, management, and evolution. There is a need to move towards a declarative model-driven methodology in distributed systems design and implementation and move away from existing imperative approaches to building distributed systems. Such modelling methodologies allow us to declaratively describe requirements that span the lifecycle of a distributed system such as capabilities, knowledge, information structures, legal terms, and processes.

For example, a model could be created to represent the legal terms necessary when writing a legally binding contract between partners in a Virtual Organisation (VO). A machine-processable vocabulary could be created to automate the task of authoring and processing digitally signed contracts. Such a contract could be used by appropriate middleware to automatically monitor the partners’ actions for

compliance with its terms. Runtime environments could also allow the evolution of a VO through the dynamic re-negotiation and modification of the legal contract. In order to achieve this level of automation, the designers should not have to imperatively describe all the necessary steps but, rather, they should only have to declaratively state the terms and conditions of their engagement in the VO; allowing the underlying architecture and infrastructure to manage the deployment, configuration, and runtime processes related to legal contracts.

If such declarative models are to be used as building blocks, they need to capture widely agreed concepts with well-understood and well-defined semantics. We therefore promote the idea of a set of semantic models for building large-scale, distributed e-Science and Grid applications. These would systematically describe and share a service’s functionality and capabilities. Conceptually, they could make use of the underlying OGSA-defined set of services and the Web Services infrastructure as illustrated in Figure 2. We expect that a core set of semantic models will emerge that span application domains. Such models would be standardised and, as a result, supported by quality tooling and infrastructure. Examples of these models are included in the rest of this section. However, this is not intended to be a complete set: it is likely that different application domains will define their own semantic models to address unique requirements.

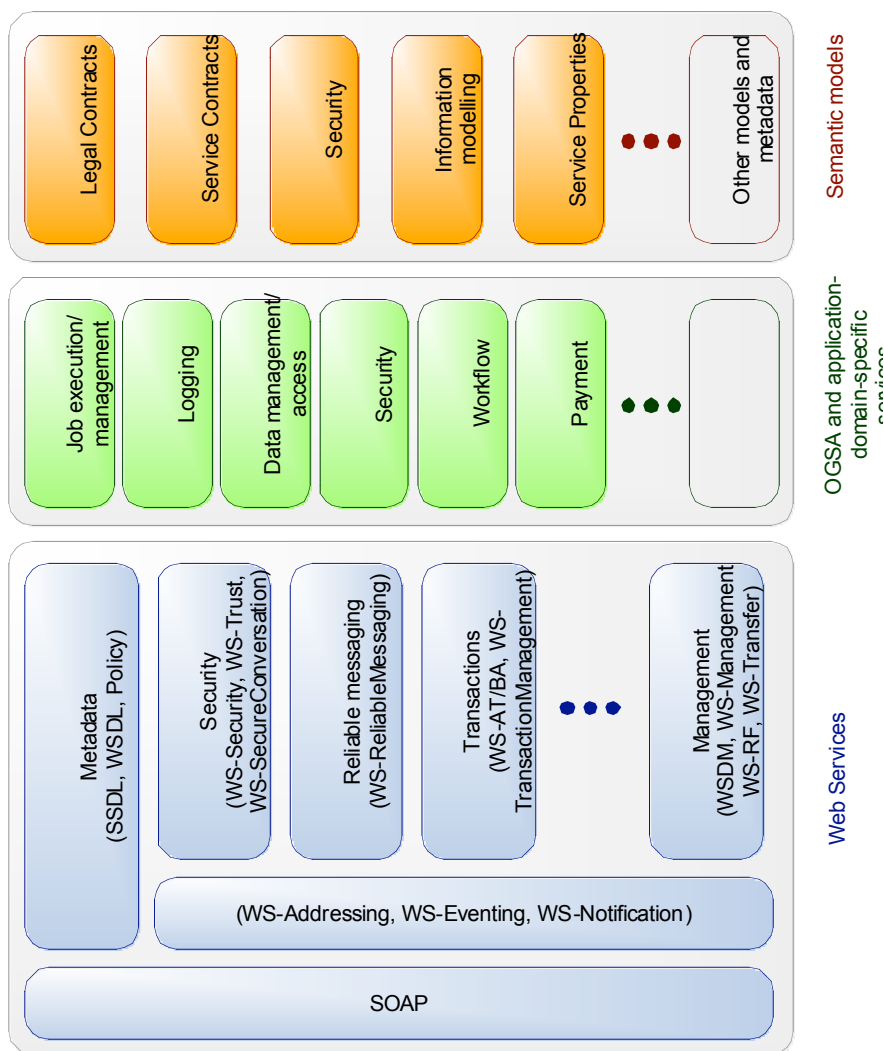


Figure 2: Modelling for service-oriented applications

4. Models

As described in the previous section, we envisage that the next-generation of large-scale service-oriented Grid applications should be implemented following a model-driven approach. The vision is one where Grid applications are “configured” from a set of services using simple and declarative languages, and with appropriate tool support. The aim is that users should require little or no knowledge about the underlying distributed computing platform.

A model provides a high level abstraction and platform independent representation of a set of related service properties. A set of models will be needed to realise the model-driven approach and they describe a service’s functional and non-functional properties including: security; static (e.g. memory and number of CPUs) and runtime (e.g. load) properties; legal contracts; and externally visible process model describing the functionality a service exposes to the network. The model-driven approach aims to enable end-users and developers to work at a higher level of abstraction than today, simplifying the development, maintenance and evolution of Grid applications.

Models need to be as *simple as possible but not simplistic*³, which is extremely difficult to achieve. A model needs to be sufficiently expressive to describe the essential properties of a service. Simple properties should be described simply while more complex properties require a proportionally greater effort.

The models will cover a spectrum of service properties. In the rest of this section, we briefly discuss models for legal contracts, service contracts, security and management properties. These are examples of some of the semantic models needed to realise the vision and others will be required. The discussion of models is technology independent, but having a single set of simple, usable and unambiguous standards is crucial in achieving interoperability. Consequently, Section 5 discusses the current state of the art for implementing models, and the Semantic Grid as a future alternative.

4.1. Legal Contracts

A legal contract model defines a shared vocabulary and protocols used by Grid participants (providers and consumers) to create, negotiate, manage and exchange agreements. The shared vocabulary is independent of the business domain and agreements should be legally binding. Writing declarative contracts using a shared vocabulary is of importance when composing services, as contract terms must have a shared meaning between the participants. The shared vocabulary should be based on terminology currently used in legal documents and specifies the promises and obligations of each participant.

Contract negotiation may be automated, semi-automated or negotiated manually. They can specify quality of service requirements, security guarantees, constraints on start and completion times, and pricing with conditions for bonuses and penalties. The GRAAP working group in GGF’s Compute Area [14] is currently defining a contract model for computational Grids. Digital representation, dynamic negotiation and runtime monitoring of contracts is also an active area of research [15].

We envisage contracts will be one of the basic building blocks in commercial Grids. Providers and consumers of services will be able to engage commercially once there are standards for exchanging and managing legally binding contracts. Contracts do not necessarily have to be in a digital representation. However, a digital representation with appropriate authoring tool, a common contract model and runtime monitoring would provide greater automation for supporting the complete lifecycle of dynamic virtual organisations. Runtime tools for monitoring and managing contracts can detect when contract conditions have been violated and react according. For example, participants should be notified of delays and the final invoiced amount reflects the agreed penalty fees specified in the contract. However, some violations will need to be resolved manually and in the

³ Quote from Albert Einstein

worst case in court. In other words, it is impossible to automate handling of every possible breach in an agreement.

4.2. Service Contracts

Service contracts describe a service's functionality that is exposed to other networked services without exposing its internal workings. The implementation details remain private to a service. Grid applications are composed from a set of service contracts and the concept of "contract first" development of Grid applications is dependent on metadata describing a service's functionality.

As service-oriented applications become more complex, it is necessary to be able to precisely and clearly describe a service's contract. A computational service may specify in its service contract that a user first authenticate themselves; stage the input data; send the binaries to the service; start the execution; and then retrieve the results.

The interactions that can take place between the services in a Grid application can be described as a "process". CSP [16] and π -calculus [17] are both declarative languages with formal semantics for the specification of processes. Formal methods tools, such as model checkers and other tools assist the development of Grid applications by identifying potential service incompatibilities – for example, services will not deadlock and the message formats are compatible.

A service contract can also be described by message exchange patterns (MEPs) [18] and is much simpler than CSP and π -calculus based languages. However, MEPS does not support the description of complex message exchange patterns. Consequently, tooling support for building applications would be limited. Whilst it would still be possible to test for message format compatibility, it would not be possible to verify where a distributed application composed from a set of services will deadlock.

Service contracts can also be described by free text in which case all aspects of compatibility need to be determined manually. If service contracts were not even specified in free text then developers would have to resort to a primitive "trial and error" methodology for developing Grid applications.

The more detailed and structured the metadata the more (development and runtime) tools can automate aspects of developing, managing and evolving an application through its lifecycle. However detail and structure come with a price in that it can be expensive to create and maintain the metadata. There is a trade-off between rich, detailed and highly structured metadata providing greater automation and less detailed and structured metadata with less automation. As a result, finding the right balance is often difficult.

4.3. Security

Providing secure access to services is one of the fundamental requirements for building large-scale distributed applications especially when services belong to different administrative domains and there is limited trust across the domains.

Security models express role-based and time-based access policies, and enforcement of integrity and confidentiality of exchanging messages. The policies are shared across administrative domains and applications and define how resources are shared in a virtual organisation. Policy decision points evaluate access requests to determine if authorisation should be granted while policy enforcement points enforces the decisions of the policy decision points.

The transition towards a semantic model makes it possible to express the roles, role-based and time-based access policy in a declarative semantic based language, simplifying and reducing the costs of managing secure access to services and Grid resources.

An example of a security policy in the medical domain is access control of patient records. People with roles "doctor" and "nurse" are permitted to update the medical sections of a patient's record, administrators can only update a patient's personal details, and patients can only view their own records. A more interesting access policy is that medical researchers can access patient records as

long their privacy is preserved meaning a patient's identity is not identifiable. A security policy language defines the rules on who is permitted to access a service, and the type of access permitted. It has to be capable of expressing commonly required access policy and needs to be flexible without compromising usability. A fine-grained security model will permit IT system administrators to define fine-grained access control to services while a course-grained model tend to be simpler for specifying access policies and managing them at runtime.

The GGF Security Area and the Web Services community has a number of working groups developing security models, including XACML [19] and SAML [20].

4.4. Service Information Model

A model-driven approach for building large-scale distributed systems require descriptions of both static and runtime properties of a service such as available number of processors and memory, operating system, software licenses, current load, and availability. Resource brokers and schedulers can match and schedule jobs on computational services and achieve greater utilisation and fault-tolerance.

Models already exist for describing services. They include CIM [21] and the GLUE schema [22]. CIM is a very detailed model while the GLUE schema is course-grained. Having multiple models introduces interoperability problems especially when it is impossible to define a mapping between the two.

CIM presents a very detailed model that can describe a system, device, network, application, database and even a physical environment. The effort required to define coherent models at this level of detail should not be under estimated.

4.5. Information Modelling

Architects and developers of e-Science applications in different application domains are often faced with similar problems of sharing information. While at the infrastructure level such problems are addressed by using widely accepted standards, production quality tooling and documentation (e.g. communications, security, reliability) the same cannot be said about the way information is represented and shared. For example, a lot of effort has been put into defining an information model for in-silico bioinformatics as part of the ^{my}Grid research project. The model captures concepts like 'project', 'user', 'experiment', 'workflow' and their relationships. The model is general enough, however, to be useful to other e-Science application domains these models need to be explicitly defined and sharable with other e-Science projects. This is currently being adopted and applied to the GOLD project.

The task of creating a model to capture information structures is difficult and time-consuming. Therefore, it would be beneficial if the e-Science and Grid communities agreed on information models that could then be applied in different application domains. Tooling to help with the process of authoring, customising, evolving and deploying such models in order to meet the requirements of an application domain will be necessary. Also, some of those models may be coupled with the use of specific services which must be supported by the deployed service-oriented architecture (e.g. a 'user' accessing a 'project' must first be authenticated and then authorised).

4.6. Summary

In this section we have given a brief overview of some of the types of models that will be needed in the declarative model-based approach for building distributed applications. This is not intended to be a complete list: there are other generic and domain-specific facets of behaviour that could be described through models. For example there is interest in the use of metadata to associate dependability information with services, allowing, for example, applications with predictable reliability to be built from sets of services [23].

It is important to note that developing and maintaining these models are difficult tasks; for example the Distributed Management Task Force have expended many person-years of effort in specifying CIM. Defining and achieving consensus requires dealing with both technical and political issues. Finding the right abstraction and balance of what should and should not be in a model, and what extensions should be supported is difficult. Political issues and cultural differences are sometimes more difficult to resolve than technical ones.

5. Metadata

Metadata is “data about data” and describes information about a physical or virtual object – for example, make, model, number of doors, recommended retail price of a car are all metadata about a car. Card catalogues in libraries are metadata to assist librarians to manage their collection and for users to discover and find books. The models described in Section 4 are all metadata about services in a distributed system. The discussion in this section focuses on technologies for the management of metadata.

Security model describes who can access a service; service contract describes the functionality a service exposes to the network; and legal contracts describe the obligations of each service in a distributed application.

It would be extremely difficult to build a grid application without metadata as it would be difficult to discover services and interoperate with them. This can lead to applications being developed using a “trial and error” approach, and existing services that could be potentially reused being re-developed.

Each model discussed in the previous section is metadata that describes a particular facet of a service. In this paper, we propose that using Semantic Web techniques to realise these models would be beneficial to the Grid community in the design and implementation of large-scale service-oriented distributed applications.

The model-driven development of service-oriented distributed systems needs a systematic approach for managing metadata. Section 4 introduced some of the models required in a model-driven approach while this section focuses the discussion of how models can be supported using the current state of the art technologies and potential future semantic Grid technologies. The later provides a more systematic approach for the governance of metadata.

5.1. State of the Art

Current support for models and metadata in Grid applications are still primitive. Metadata is often embedded in code libraries, free text documentation and XML documents, or a combination of all three. The vocabulary used in the metadata is often ambiguous and is difficult, and at times impossible, to share between services.

Without any form of metadata, it is extremely difficult to provide tooling to automate aspects of the development, runtime management and evolution of distributed applications. All knowledge regarding configuration and access policy are undocumented. At best, the knowledge is private to administrators. Developers and users may have to “ask around” in the hope of finding out how to gain access to a service. Without metadata, the process of forming, maintaining and evolving a distributed application is impractical.

Free text documentation that explicitly records the knowledge embedded in code libraries and by administrators is metadata. The documentation provides the means for users to find details on how to create, maintain and evolve distributed applications without the need to “ask around”. Search engines and free text search can help users find relevant information. However, the process of forming and maintaining Grid applications is a completely manual process since the metadata is unstructured, not machine-processable and often ambiguous. Free text documentation is much more desirable than the situation where there is no metadata. However, it is still impractical for building and maintaining Grid applications.

Ambiguity is removed when metadata is expressed using terminology from a controlled vocabulary (ontology), so making it more sharable. An ontology is the specification of a set of concepts and their relationships, and they can formally and declaratively specify the models described in Section 4.

Metadata are machine-processable when both the metadata syntax (structure) and vocabulary is standardised. XML Schema is the commonly accepted language for specifying models – that is, the Web community is using XML Schema as their “ontology” language even though it does not have formal semantics – for example, XACML [19] defines an authorization and entitlement policy ontology, WSDL [4] and BPEL abstract processes [24] [25] define service contract ontology.

Once metadata is machine-processable, the potential benefits are great. Users can discover services with far greater ease due to improved search precision and recall, and services will be able to interoperate. The metadata makes it possible to have rich tools for the development, runtime management and evolution of Grid applications.

However, specifying the structure and vocabulary using XML schema is not ideal. It is a relatively low-level language for defining models and constrains the model to a hierarchical structure. Higher-level languages would be more effective for the specification and maintenance of models.

Also, there is currently no systematic way to exchange metadata between services at this time. Instead, ad hoc protocols are used. A systematic approach that supports the explicit handling and delivery of metadata would advance the current state of art for Grid systems and a path towards realising the Grid vision – sharing of diverse services in a flexible, interoperable, secure and reliable distributed environment across administrative domains.

5.2. Semantic Grid

Semantic Grid is supported by two key building blocks – semantic web technologies for creating and maintaining models (ontologies), and semantic aware services and protocols for exchanging metadata. Semantic Grid is an extension of Grid and provides the infrastructure that systematically manages the complete lifecycle of metadata.

The Semantic Grid aims to provide an infrastructure within the Grid middleware – providing a core set of services and protocols to support the sharing and management of all aspects of metadata. This will enable unanticipated reuse of Grid services and resources, better support for interoperability, and flexible Grids.

5.2.1. Semantic Web

There is currently great interest in the Grid community to leverage Semantic Web technologies for Grid computing. The vision is that Semantic Web and other related technologies could greatly reduce the effort required to develop, maintain and evolve future e-Science and Grid applications.

Semantic Web technologies use high-level ontological languages such as OWL to define models and provide generic software for the provisioning and processing of metadata, including reasoning capabilities. Reasoners derive implicit knowledge (facts) from metadata [26] that will simplify the architecture, implementation and management of future e-Science and Grid applications.

The formal semantics that underpins ontological languages makes the metadata machine-processable with a shared semantics. The two main Web knowledge representation standards are RDFS (Resource Description Framework Schema) [27] and OWL [28] and both are W3C Recommendations.

RDFS provides a basic vocabulary to express conceptual models while RDF provides a mechanism to state facts (metadata) about a resource. RDF provides the foundations to support ontology languages and it does not necessarily have to be defined in RDFS since it provides a syntax for the exchange of metadata on the Web. RDF is simply a graph consisting of RDF triples. Each triple defines a *subject*, *predicate* and an *object*. These subjects, predicates and objects literals may not be defined in an

ontology. However, there is no shared semantics without an ontology and the processing of the metadata may be ambiguous.

OWL is a Web language for specifying ontologies (knowledge domains). The formalism is the means by which reasoners can extract and derive implicit knowledge. There are graphical tools support such as Protégé [29] for domain experts to specify an ontology without needing to understand the underlying OWL syntax in XML. Graphical tools are just as important as the underlying formal semantics. It is realistic for domain experts to define ontologies using user friendly GUIs but unrealistic for them to directly write their ontologies in OWL. Further, semantic Grid will need to provide generic components for the provisioning, processing and exchanging of metadata between autonomous services.

OWL-DL, one of the three species of OWL, utilizes description logic (DL) reasoners such as Fact++ Racer, KAON2 and Pellet, to reason about knowledge. Description logic supports a decidable fragment of first order logic. From the architect and developer's view, a DL reasoner is a generic component that provides reasoning capabilities independent of the application domain.

5.2.2. Semantically Aware Services and Protocols

The Grid currently deals with semantic information (metadata) in an ad-hoc manner thus providing poor mechanisms for sharing and processing of metadata. Understanding and knowing how to retrieve metadata from a service is embedded in best practices and experience. The sharing, customisation and adaptation of metadata is difficult, and dependent on scarce and expensive human effort resulting in systems becoming brittle during evolution.

A service has semantic capabilities if it can provision both static and dynamic metadata in the same sense as a data grid service provisions data. The semantic grid will provide services to support the creation, storage, retrieval, query, update, and deletion of semantic information, which can then be exploited to deliver extra functionality.

Finally, the semantic Grid defines a set of Grid protocols, standards and core semantic services to support the exchange of semantic information between services. There are three essential services – ontology services that store domain models (ontologies), instance stores that store metadata and reasoning services. The protocols define how semantic information are exchanged between an ontology service, reasoner and instance store to support a more secure, flexible, controlled, co-ordinated and reliable Grid. For example, if the semantic information is exchanged by reference then a service processing the semantic information needs to know how to access the information from an instance store; if the semantic information is exchanged by value then the service needs to know how to extract the information from the messages.

The reference architecture Semantic OGSA (S-OGSA) is defined in [30]. S-OGSA defines the systematic management of semantic information in OGSA, core capabilities of services in S-OGSA and how to evolve existing OGSA services into S-OGSA services. This paper starts to address both the issues described in this paper and in [2] – “Future For European Grids: GRIDs and Service Oriented Knowledge Utilities (SOKU)”.

Section 6 discusses how realistic it is to develop Grid applications using Semantic Web technologies. The discussions are based on experiences from current and past e-Science projects at ESNW (e-science North West Centre) and NERESC (north-east regional e-science centre).

5.2.3. Open Issues

Realising the vision of the semantic grid requires a number of key questions to be investigated, including: are semantic web technologies scalable; is it realistic to define domain models in ontologies; what's the impact of a semantic approach to legacy grids; how do we minimize impact of migrating to semantic capable services; what are the minimum knowledge services need; what should be their capabilities; how do we harvest and tend the semantic content; is there content that is common for

all Grids and how much is application specific; how, when and where does a semantic approach add value to a “traditional” Grid approach; what is an architectural framework for a Semantic Grid?

6. Case Studies

The UK e-Science programme has funded a number of research projects in the area of e-Science and the Grid. We briefly present the experiences of three such projects, focusing the discussion on the relationship each has with the model-driven approach and the semantic Grid. We then outline an example of a proposal to define a set of services in one domain, which could benefit from this approach.

6.1. GOLD

GOLD is a collaboration between the Universities of Newcastle and Lancaster [31]. Its focus is the development and deployment of the necessary infrastructure to facilitate highly dynamic Virtual Organisations (consortia). The chemical engineering sector provides the use case that drives the project.

Security, trust, process and information modelling, and VO management are some of the key areas of interest to GOLD. A decision was made early in the project to use, where possible, standard, interoperable technologies and existing tooling.

It is assumed that those creating and managing VOs will not be programmers. Instead, a declarative approach has been chosen to express the business processes, security policies, legal contracts, and all other aspects that make a VO through wizard-based tooling. Users are able to input their requirements and design their processes. It is the responsibility of the middleware platform to “process” the declarative descriptions in order to configure the necessary infrastructure amongst the collaborating partners, monitor all the interactions, and manage its runtime.

GOLD has delivered an ontology for legal contracts, enable participants to exchange electronic contracts and for services to negotiate and monitor contracts. Nested contracts are supported in GOLD to manage highly complex contracts.

6.2. ^{my}Grid

The ^{my}Grid project provides a Grid infrastructure for bioinformaticians to conduct *in silico* experiments [32]. Life Sciences researchers search, analyse, manipulate and link datasets to perform data intensive experiments. The infrastructure software from the ^{my}Grid project is free for download and has been used by collaborating scientists for building workflows into the investigations of Williams-Beuren Syndrome and Grave’s Disease.

Discovery of data and services that analyse and manipulate data, manage provenance information, and workflows are the core ^{my}Grid components. Workflows represent a life scientist’s experiments and its enactment is the act of performing the experiment. Life scientists first need to discover, on the grid, the relevant data sources and services when defining their workflows. Provenance records the details of the experiments so results can be validated and verified.

The task of finding relevant data and services can be time consuming since there are thousands of services and data providers on the grid. The numbers will continue to grow as e-Science becomes more established in the science community. Discovery based on searching for keywords in free text documentation (unstructured metadata) is imprecise and scientists will find it too time consuming, especially as more and more datasets and services become available.

Semantic search engines offer scientists greater precision and reduce the effort required in finding data repositories and services for their workflows. Descriptions of available data and services are annotated with vocabulary from an ontology and the semantic search engines rely on these

annotation to provide greater precision in search. Providers will have tooling support for annotating their datasets and services.

A workflow description represents an experiment and its enactment engine runs the in silico experiment. Tooling and runtime environment hide the complexity of the underlying grid middleware for ^{my}Grid users. Existing workflow tooling and enactment engines already exist today and they rely on process models such as π calculus [17] to represent and store workflow specifications.

Scientists protect their workflows as they represent their intellectual property and would only share their workflows with trusted colleagues. Workflow repositories can store these workflows for future reuse but should, therefore, only be accessible by authorized personnel.

Results from experiments need to be verifiable. Provenance records the data, methods and results from in silico experiments. Provenance records can be systematically collected by workflow enactment engine, resulting in large quantities of provenance data. Semantic search technologies will again be required to verify past experimental results.

Provenance also needs to be protected from unauthorized access since it is trivial to reverse engineer workflows from provenance. However, like the workflow specifications, scientists would want to share their records amongst trusted collaborators.

Specifying, managing and monitoring access rights to workflows and provenance records is required in ^{my}Grid even though the data is publicly accessible. Tooling, management and monitoring of access rights are required in the formation and management of dynamic virtual organisations for ^{my}Grid and other grid projects.

The ^{my}Grid project has demonstrated the feasibility of developing complex domain ontologies using Semantic Web technologies. It also provides the experience required for defining appropriate models for virtual organisations.

6.3. RealityGrid

RealityGrid is a computational Grid system where scientists interactively steer, in real time, a simulation (in-silico experiment) in search of interesting regions of a parameter space. It provides the experience and use cases for defining models for VO, contracts and processes. A simulation requires the formation of a virtual organisation and sharing of resources (e.g. processors, storage, visualisation devices and Access Grid rooms) for geographically dispersed scientists to collaboratively steer an experiment.

Fault tolerance in RealityGrid is supported by checkpointing. It stores a snapshot of the state of a simulation to disk and use the snapshot to restart a simulation in the future. Checkpointing allows a scientist to restart a simulation from the snapshot after a system failure, support "what if" scenarios and recording the parameter searches to validate and verify results. The metadata include who, when, what and where attributes, plus additional information specific to the application domain. Checkpoints are very similar to provenance in ^{my}Grid and can also be used to verify and validate experimental results.

Checkpointing data, like provenance, needs security. Without security, it would be straightforward for competing researchers to gain access to protected experimental data. Policies need to be defined by the collaborating scientists to control access to checkpointing data.

Ideally, service providers advertise their resources through directory services describing their offerings via policies. Policies describe a broad range of properties including capabilities, quality of service, pricing, availability and management. Models and tooling are required for creating and submitting policies to directory services, and for scientists to efficiently locate required resources. Contracts, specifying the terms and conditions such as service level agreements and pricing, are then negotiated between the consumers and providers, contracts are then validated and monitored at

runtime. In RealityGrid, these tasks are currently either done manually (e.g. sending emails and making phone calls) or are not supported.

6.4. JISC eLearning Framework

The JISC eLearning Framework (<http://www.elframework.org/framework>) defines a set of services for supporting the requirements for eLearning. It includes all aspects of allowing end users to discover, access, use and publish information in learning and research activities. The identified services include discovery and access to scholarly journals, textbooks, images, computational resources, collaborative environments (e.g. AccessGrid and chat rooms), digital imagery, calendar, reporting, curriculum, and grading.

The eLearning framework has a layered architecture. A set of common services (basic building blocks) is in the lowest layer, supporting application dependent services. Best practices (practical blueprints) of using SOA for building eLearning systems are also part of the framework.

Frameworks such as this, comprising a set of related services that are intended to be utilised to build applications, would be key beneficiaries of the approach proposed in this paper. The services could be described in a consistent manner, making them easier to discover, interpret, compose and integrate.

7. Conclusions and Recommendations

Grid aims to support secure, flexible, open, interoperable and coordinated resource sharing by providing a middleware platform for distributing computing. This paper argues that future Grid applications should be implemented by following a declarative semantic model-driven approach to realise the Grid vision, one in which non-IT experts can configure and maintain their Grid applications.

Simple models with sufficient expressiveness need to be standardised. Examples include access policy, legal contracts, service contracts, information models and many others. Only through experience in building large-scale service-oriented distributed application will the Grid community find the right abstractions for declarative and semantic models.

The Semantic Grid provides generic components, services and protocols in support of the declarative model-driven approach. It aims to provide higher-level languages, tools and reasoners for the specification of models, provisioning and consumption of knowledge (metadata), and generic reasoning capabilities. These include model repositories, instance stores that store metadata and reasoning services.

While there is existing work that assists in meeting this goal, we recommend further research and development in a number of areas including:

- Well-defined, simple, composable, and unambiguous semantic models. Each model describes a particular functional or non-functional characteristic of a service;
- Development and runtime tooling to support metadata creation, publication and discovery, semantic composition, decomposition and behaviour prediction.
- Tooling for distributed application lifecycle management including runtime monitoring and support for evolution.
- Documenting best practices and patterns for building, deploying and managing Grid applications that utilises the semantic model based approach advocated in this paper.

Acknowledgements

The authors wish to thank Pinar Alper, John Brooke, Adrian Conlin, Michael Parkin, Ellis Solaiman and Jim Webber for discussions and comments on this report.

References

- [1] GGF, "OGSA Working Group," <https://forge.gridforum.org/projects/ogsa-wg>.
- [2] "Future for European Grids: GRIDs and Service Oriented Knowledge Utilities. Vision and Research Directions 2010 and Beyond," January 2006.
<http://cordis.europa.eu/ist/grids/ngg.htm>
- [3] W3C, "SOAP 1.2 Part 1: Messaging Framework," M. Gudgin, M. Hadley, N. Mendelsohn, J. J. Moreau, and H. F. Nielsen, Eds. <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>; W3C, 2003.
- [4] W3C, "Web Services Description Language (WSDL)." <http://www.w3.org/2002/ws/desc>.
- [5] OASIS, "WS-RF Specifications," OASIS, 2004.
- [6] OASIS, "Web Services Notification (WS-Notification)." <http://www.oasis-open.org/committees/wsn>.
- [7] J. Alexander, D. Box, L. F. Cabrera, D. Chappell, G. Daniels, A. Geller, R. Janecek, C. Kaler, B. Lovering, D. Orchard, J. Schlimmer, I. Sedukhin, and J. Shewchuk, "Web Service Transfer (WS-Transfer)." <http://msdn.microsoft.com/ws/2004/09/ws-transfer/>, 2004.
- [8] D. Box, L. F. Cabrera, C. Critchley, F. Curbera, D. Ferguson, A. Geller, S. Graham, D. Hull, G. Kakivaya, A. Lewis, B. Lovering, M. Mihic, P. Niblett, D. Orchard, J. Saiyed, S. Samdarshi, J. Schlimmer, I. Sedukhin, J. Shewchuk, B. Smith, S. Weerawarana, and D. Wortendyke, "Web Services Eventing (WS-Eventing)."
<http://msdn.microsoft.com/webservices/understanding/specs/default.aspx?pull=/library/en-us/dnglobspec/html/ws-eventing.asp>.
- [9] OASIS, "Web Services Distributed Management." http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm.
- [10] OASIS, "Web Services Security (WS-Security)." <http://www.oasis-open.org/committees/wss>.
- [11] OASIS. <http://www.oasis-open.org>.
- [12] W3C. <http://www.w3c.org>.
- [13] WS-I. <http://www.ws-i.org>.
- [14] GGF, "WS-Agreement (GRAAP WG)." <https://forge.gridforum.org/projects/graap-wg>.
- [15] C. Molina-Jimenez, Santosh Shrivastava, E. Solaiman, and J. Warne, "Run-time monitoring and enforcement of electronic contracts," *Electronic Commerce Research and Applications*, vol. 3, pp. 108-125, 2004.
- [16] C. A. R. Hoare, *Communicating Sequential Processes*: Prentice Hall International Series in Computer Science, 1985.
- [17] R. Milner, *Communicating and Mobile Systems: the Pi-Calculus*: Cambridge University Press, 1999.
- [18] W3C, "Web Services Message Exchange Patterns." <http://www.w3.org/2002/ws/cg/2/07/meps.html>, 2002.
- [19] OASIS, "Extensible Access Control Markup Language (XACML)." <http://www.oasis-open.org/committees/xacml>.
- [20] OASIS, "Security Assertion Markup Language (SAML) v2.0." <http://www.oasis-open.org/committees/security>, 2004.
- [21] DMTF, "<http://www.dmtf.org/standards/cim/>." <http://www.dmtf.org/standards/cim/>.
- [22] HENP Intergrid Joint Technical Board, "GLUE Schema." <http://www.hicb.org/mailman/listinfo/glue-schema>.
- [23] J. S. Fitzgerald, S. Parastatidis, A. Romanovsky, and P. Watson, "Dependability-explicit computing in Service-oriented Architectures " presented at Supplementary Volume of 2004 International Conference on Dependable Systems and Networks. , 2004.
- [24] OASIS, "OASIS Web Services Business Process Execution Language." <http://www.oasis-open.org/committees/wsbpel>.
- [25] J. Webber and M. C. Little, "Introducing BPEL4WS," *Web Services Journal*, vol. 3, 2003.
- [26] W3C, "Semantic Web." <http://www.w3.org/2001/sw/>.
- [27] W3C, "RDFS." <http://www.w3.org/TR/rdf-schema/>.
- [28] W3C, "Web Ontology Language (OWL)." <http://www.w3.org/2004/OWL/>.

- [29] Stanford University, "Protege." <http://protege.stanford.edu/>.
- [30] 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*, vol. 4, pp. 102-115, 2006.
- [31] Universities of Newcastle and Lancaster, "Gold Project." <http://www.goldproject.ac.uk/>.
- [32] myGrid Project team, "myGrid." <http://www.mygrid.org.uk>.