

# Semantic Grid Infrastructure for Applications in Biomedicine

Martin KUBA<sup>1</sup>, Ondřej KRAJÍČEK<sup>1</sup>, Petr LESNÝ<sup>2</sup>, Tomáš HOLEČEK<sup>3</sup>

<sup>1</sup>*Institute of Computer Science,  
Masaryk University Brno  
Botanická 68a, 602 00 Brno, Czech Republic  
makub@ics.muni.cz, krajicek@ics.muni.cz*

<sup>2</sup>*ENT Department of Faculty Hospital Motol  
V Úvalu 84, 150 06, Praha 5 – Motol, Czech Republic  
petr.lesny@lfmotol.cuni.cz*

<sup>3</sup>*FMS FHS Charles University  
U Kříže 10, 158 00 Praha 5 Jinonice, Czech Republic  
holecek@ojrech.cz*

**Abstract.** In this paper, we present a prototype design of a semantic adaptive grid infrastructure called SEAGRIN, based on a decentralized execution of workflows. The design of the infrastructure is motivated by the needs of a biomedical grid. We provide an overview of current semantic grid technologies along with discussion of their suitability for this particular purpose. We address some of the shortcomings of existing solutions in our proposed infrastructure.

**Keywords:** semantic, grid, ontology, integration, medical, services

## 1 Introduction

Expanding knowledge base of biomedical domains is reflected in growing volumes and complexity of clinical data generated and utilized in contemporary health services. Development of software for interpreting the medical data (historically named "the medical expert systems") goes forward together with the development of systems for storing and retrieving the data ("the medical information systems"). Whereas the medical information systems are widely utilized for managing hospital data and basic communication between public health service subjects, the development of systems, which are capable of complex data evaluation in medicine, meets serious methodological and computational difficulties. Some of the difficulties can be overcome by applying semantic grid technologies. We present an overview of the technologies and our semantic grid architecture.

## 2 Semantic Grid

### 2.1 Grids

The original idea of *computational grid* as presented in [5] envisioned a pervasive infrastructure for sharing of high-end computational resources. That appealed mostly to High Performance Computing (HPC) community, where the demand for large number of CPUs, petabytes of disk storage and gigabits-per-second networks is justified. However, research focus shifted over time from the high-end resources part of the definition to the part about resource sharing. Thus two new types of grids emerged [2]. *Collaborative Grids* enable collaboration of geographically distributed groups of individuals using tools like video conferencing and remote control of instruments. *Information Grids* enable information, data and knowledge sharing across organizational boundaries.

The advent of OGSA (*Open Grid Services Architecture*) based grid infrastructure on web services, which in turn are based on XML and other open standards. That allows really interoperable and uniform access to all grid services. However, even OGSA-compliant grid services are described on syntactic level only. To find and successfully use a particular grid service, one must know exact character strings used as names for the service, its operations and data types. The knowledge of the *semantics* (i.e. meaning) of the data and operations is not expressed explicitly in machine-understandable form. It is only hard-coded in programs written by human programmers [9]. Thus composing such grid services into more complicated workflows is a hard human work.

Here come to a rescue the closely related ideas of *semantic grid* [14] and *semantic web services* [3]. They are both based on technologies developed for the *semantic web*. While the semantic web “is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation” [1], the semantic grid and semantic web services want to give the well defined meaning to services, as opposed to static web documents.

### 2.2 Ontologies

The well-defined meaning is introduced using *ontologies* [10]. Ontologies are a concept developed in the area of artificial intelligence (AI) and are used to capture knowledge about some domain of interest. Ontology is an explicit formal specification of how to represent the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them. In other terms, ontology is a *taxonomy* (classification structure) combined with inference rules.

Recently W3C defined a Web Ontology Language (OWL) [7], based on XML, which allows describing ontologies in a standard way. Ontologies written in OWL consist of classes, properties and individuals, all identified via URIs (*Uniform Resource Identifiers*). Classes are sets of individuals, described either by enumerating class members, or by a set of conditions which members must satisfy. Properties are binary relations on individuals, i.e. link two individuals together. Properties can be

defined as inverse, transitive, symmetric or functional (single valued). Individuals may belong to more than one class.

One important feature of OWL is that ontologies can be composed, i.e. there is no need to have one large ontology, which comprises everything. Ontologies can be integrated from smaller parts, thus enabling local management of knowledge.

Three versions of OWL are defined: Lite, DL and Full. The DL version is based on *Description Logic*, which provides decidable reasoning, thus enabling the existence of reasoners (e.g. RACER [13]) – software engines, which can deduce relationships present in ontology but not explicitly written. Reasoners can also uncover previously unavailable information after joining several local ontologies into one [6].

Once some community agrees on ontology, meaning can be expressed by references to entities in that ontology.

### 2.3 Semantic Web Services

Traditional web services are described using WSDL (*Web Service Description Language*) on a syntactic level, as collections of operation names and XML Schema data types. This description can be enriched by adding semantic information. There were identified at least 4 areas where ontologies can be used when semantically describing web services [3]:

- Data Semantics – it defines meaning of the data, i.e. input and output data of operations,
- Functional Semantics – it defines meaning of the operations, i.e. how they transform input data to output data,
- QoS Semantics – it provides meaning for *Quality of Service* aspects, such as price, availability, level of trust, etc. Service discovery may be influenced by these characteristics.
- Execution Semantics – it provides details like preconditions, effects and conversation patterns of service invocation.

The relation between web service characteristics and concepts in some OWL ontology can be expressed in formal ways, either by annotating the WSDL with references to URI in the ontology, or by an external description. There are some efforts to standardize this semantic annotation of web services, namely OWL-S (OWL for Services) [12] proposed by OWL-S Coalition and WSDL-S [15] proposed by METEOR-S project.

Once a web service is semantically annotated, this information can be advertised in some semantically-augmented service registry, and that information can be used for service discovery. Current non-semantic searches in registries like UDDI use simple keyword matching and the search precision is inadequate. Semantically enhanced searches can provide much better search precision. A nice example is in [8]:

When searching for a service which "returns a Quote for a Hard Drive", using non-semantic search for keyword `getQuote` returns far too many results, while search for more specific `getQuoteForComputerHardDrive` may miss services with different wording like `getHardDriveQuote`.

When a semantic search for concept `ComputerParts:#getHardDriveQuote` is employed, it can find the relevant services with different wording which non-semantic search missed. But even more, by using reasoning, it can find that a service annotated with the concept `ComputerParts:#getSCSIDriveQuote` is also a potential candidate, because the concept is in subsumption relation to the more general concept in the domain ontology.

## **2.4 Semantic Grid Services**

Grid services extend web services, which themselves are stateless and without a defined lifecycle, by adding *WS-Resources* as defined in WSRF (*Web Services Resources Framework*). *WS-Resources* are stateful and have a lifecycle, are created on demand and destroyed when no longer needed.

Grid services should also deal with security issues. An important feature of grid security is credential delegation, which allows a service to act on behalf of its user for some time, after it receives short-lived credentials derived from user's credentials.

The Semantic Grid is defined as "extension of the current Grid in which information and services are given well-defined meaning" [14]. As grid services are web services, it is straightforward to use technologies developed for semantic web services also for grid services.

## **3 Biomedical Grid**

Development of complex medical software requires some tools for indexing medical data; contemporary advancements indicate that these tools should be based on suitable classification systems or - more generally - on ontologies. Grid computing deals with interoperability problems in distributed environment. From this point of view, the Semantic Grid architecture is well suited for biomedical data processing:

- Semantic web resources allow us to implement a suitable ontology background, which is necessary in order to index data from various information sources and to process them in the Grid environment.
- The Grid environment is intrinsically able to deal with fragmentation and heterogeneity of the information sources, programming languages and operating systems.
- The distributed network environment suits distributed and modular nature of biomedicine (research, medical specialties and subspecialties, medical practitioners, laboratories, libraries).
- The modular structure of the Grid allows biomedical professionals (healthcare specialists, researchers) to concentrate on studied problems (and implement them as web services). It isolates them from the network infrastructure, security issues and other problems arising with developing and presenting their applications.

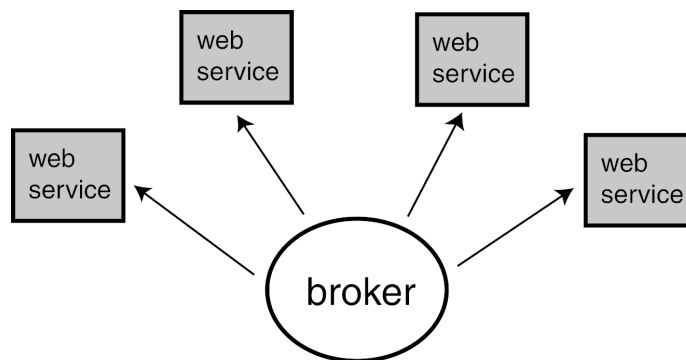
Moreover the Grid infrastructure dedicated to the biomedical data processing (The Biomedical Grid) provides foundations for creating an environment for collaboration of specialists from various biomedical areas, based on the concept of Access Grids

[4]. This environment extends the concept of audiovisual collaboration by sharing applications and – *in sensu stricto* – sharing the recorded knowledge.

#### 4 SEAGRIN - Semantic Grid Infrastructure

Our approach to designing infrastructure for semantically enhanced services is driven by the challenges in complexity of biomedicine applications. Applications (such as [11]) of Grid Computing in biomedicine are taking the usage of computing in this field to a new level of functionality.

However, the usual approaches are based on fairly centralized architecture, which employs brokers to accomplish the integration tasks. By using centralized broker architecture, new single point of failure may be introduced to the system. Using centralized approach may also present scalability problems in the future, which may prove not to be easily solved by introducing additional brokers.



Centralized broker architecture

In designing our infrastructure, we took substantially different approach. We propose an architecture called SEAGRIN (*SEmantic Adaptive Grid INfrastructure*). The design of SEAGRIN is motivated by the following goals:

- To integrate easily into existing infrastructure based on Web services, without imposing any additional implementation overhead on the existing services themselves.
- To provide robustness capabilities, such as fault tolerance, adaptation and dynamic reconfiguration.
- To incorporate dynamic changes to the system and allow integration of the SEAGRIN infrastructure with foreign systems based on Web Services technology.

- To define responsibilities of individual components comprised by the infrastructure, separate their concepts and thus aid the overall robustness of the system.

Our approach is based on Grid Computing concepts, as the name implies, mainly to take advantage of existing features typical for Grid applications. We provide architectural patterns describing the design of distributed applications based on services with service interconnection.

Our infrastructure is based on two key concepts: *wrappers* and *workflows*.

#### **4.1 Semantically Aided Integration of Services**

Traditional model of service oriented applications is based on client/server model. In the Service Oriented Architecture, the client functionality is implemented by service consumption. Client invokes one or more services and processes the results obtained from the invocations. The client must have explicit knowledge of the services.

However, the goal of our infrastructure is to allow semantically aided integration of services regardless of their origin, implementation and purpose, in decentralized way. These services, which represent the functional building blocks are denoted as *Primary Services*. To allow integration of such services there is a set of basic requirements which the primary services must meet:

- A primary service **must** be a Semantic Web Service, i.e. it **must** provide semantic annotation of its interface. These annotations enable composition of primary services into more complex structures, such as workflows.
- A primary service **must** conform to the WS-Interoperability Basic Profile 1.1.
- A primary service **should** provide additional description of their supported security and invocation mechanisms. This description is meant to be a XML document conforming to some predefined XML Schema. This will be replaced by conformance requirement to further WS-I profiles, as they become available and widely recognized.

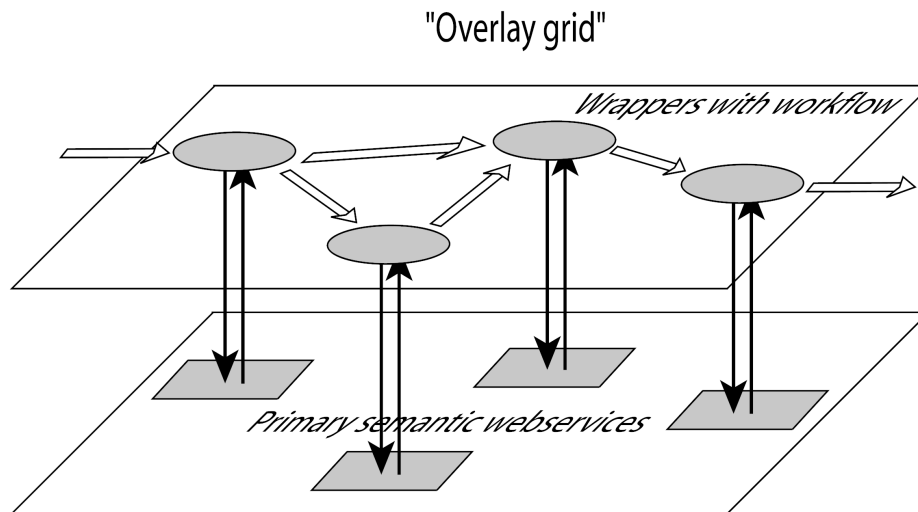
#### **4.2 Wrappers**

Wrappers provide mechanism for incorporating Primary Services into Workflows. Each primary service (i.e. a web service) is encapsulated by a specialized grid service called *Primary Service Wrapper* (or just Wrapper). Wrapper provides all interaction between the SEAGRIN infrastructure and the particular primary service. Their key purpose is to provide intermediary mechanism between the service and the infrastructure, such as translation of the input and output data (messages) so that the primary service understands them. Also, wrappers may implement other functionality, such as monitoring the service, fault tolerance, etc.

---

<sup>1</sup> The usage of the keywords: “must”, “should”, etc. should be interpreted as defined in IETF RFC 2119.

By introducing the wrapper layer over primary services, the infrastructure itself can evolve, incorporate new features and technologies without affecting the primary services and thus prevents loss of functionality, which may otherwise happen. We call this approach an **Overlay Grid** (in the same sense as in overlay network).



### 4.3 Workflows

To interconnect services, we have adopted a notion of Workflow of services. The key idea is to connect services, which implement simple, well-defined functionality to form larger systems, which are able to solve more complex tasks.

Workflows are created and managed using services as well and provide client interface in the form of services. Workflow is built from a set of grid services, each with well defined interface and responsibility. Services, which implement the workflow management and other support functionality, are called *Infrastructure Services*.

Workflow infrastructure services are grid services defined by OGSA and WSRF; actually the workflows are represented using WSRF resources. The infrastructure services are:

- **Primary Service Wrapper** – provides encapsulation of one particular primary service. All interaction with the primary service is done through its wrapper.
- **Composer Service** – builds workflows based on the description of the task the workflow is intended to solve.
- **Controller Service** – provides workflow lifecycle and status management and access to services which implement the workflow monitoring.
- **Accessor Service** – provides a client-oriented interface to the workflow. Clients use this service to submit jobs to the workflow and receive results.

- **Nest Service** – provides means for automated wrapper creation, based on the description and semantic annotation of a particular application service. The wrappers are created on demand, by providing the service description, but particular wrappers may be created in advance and/or cached for reuse. However, this behavior is implementation specific and is not mandatory.
- **Workflow Junction Service** – implements a conditional branch (a filter) in a workflow. It allows different handling of messages which are passed to it according to some predefined condition.

#### 4.4 Wrapper Implementation Aspects

We need to construct workflows of arbitrary services, thus issues regarding service compatibility arise. Naturally, the most simple solution, which allows only services following some predefined specification and “agree” on the message level, is not feasible.

Another solution, which involves implementation of some intermediary, seems practical. The intermediary applies some kind of transformation mechanism on the messages as they are transmitted between services. Such transformation mechanism must take into account not only the syntactic definition (i.e. the structure) of messages, but the message semantics as well. Moreover, the transformation mechanism must closely follow changes in existing or creation of new services. Due to the dynamic nature of the Grid environment, the implementation of the intermediary mechanism should be service-specific and created automatically, on demand, without explicit user intervention.

In SEAGRIN, the task of implementing the intermediary is taken by the Wrapper services. A nest service is used to create and deploy a wrapper service for particular primary service based on its description and semantics. The wrapper service itself then does the necessary processing of messages.

Also, additional requirements, related to the general organization and management of workflows, have been identified. Based on these requirements, several “functional patterns” have been defined for wrapper services, as follows:

- **Translator** – implements syntactic transformations of messages by applying XSLT transformation templates.
- **Converter** – implements semantic transformations (i.e. *conversions*) on messages, based on semantic annotations of primary services. These conversions do not alter the structure of data, but the data itself based on their interpretation by a particular service, for example converting units between various measurement systems.
- **Merger** – merges several incoming messages from various sources into one input message to be passed to the encapsulated primary service.
- **Splitter** – duplicates an outgoing message, passing it to specified successors.

The design of the wrapper infrastructure is still work in progress and the concept is being refined further. The proposed design may be verified using prototype implementations.

## 5 Conclusions

We have presented a design blueprint for adaptive infrastructure based on the concepts of Grid Computing, Ontologies and Semantic Services. Our approach is substantially different to the commonly adopted idea of central broker services, which tend to cumulate responsibilities. Our design is driven by the needs of biomedical applications, but we believe it will have broader range of applications. The presented concepts and infrastructure are part of the ongoing research and will be extended in the future.

## 6 Acknowledgments

This research is supported by a CESNET (<http://www.cesnet.cz>) research intent "Optical Network of National Research and Its New Applications" (MSN6383917201) and research project "MediGrid – methods and tools for Grid applications in biomedicine" (Czech Academy of Sciences, grant T202090537).

## 7 References

1. T. Berners-Lee, J. Hendler, O. Lassila: "The Semantic Web", *Scientific American* (2001)
2. I. Blanquer et al.: HealthGrid Whitepaper, <http://whitepaper.healthgrid.org/>
3. J. Cardoso, A. Sheth: "Introduction to Semantic Web Services and Web Process Composition", *Lecture Notes in Computer Science*, Springer-Verlag, Volume 3387 / 2005, ISBN 3-540-24328-3 (2005)
4. L. Childers et al.: "AccessGrid: Immersive Group-to-Group Collaborative Visualisation", In: *Proceedings of Immersive Projection Technology* (2000)
5. I. Foster, C. Kesselman : Chapter 2 of "The Grid: Blueprint for a New Computing Infrastructure", Morgan-Kaufman, 1999. <http://www.globus.org/alliance/publications/papers.php#chapter2>
6. F. Heine, M. Hovestadt, O. Kao: "Towards Ontology-Driven P2P Grid Resource Discovery to Facilitate Composition", In: *Proceedings of Fifth IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, Pennsylvania (2004)
7. M. Horridge: Protégé OWL Tutorial, <http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf>
8. P. Rajasekaran, J. Miller, K. Verma, A. Sheth: "Enhancing Web Services Description and Discovery to Facilitate Composition", In: *Proceedings of International Workshop on Semantic Web Services and Web Process Composition* (2004), <http://lsdis.cs.uga.edu/lib/download/swsrpc04.pdf>
9. M. Uschold: "Where are the Semantics in the Semantic Web?", *AI Magazine*, Volume 24, Issue 3 (2003) 25-36, <http://www.starlab.vub.ac.be/WhereAreSemantics-AI-Mag-FinalSubmittedVersion2.pdf>

*Semantic Grid Infrastructure for Applications in Biomedicine*

10. Vojtěch Svátek: “Ontologie a WWW”, In: *Proceedings of DATAKON (2002)*, ISBN 80-210-2958-7 (česky)
11. Integration Broker for Heterogeneous Information Sources (IBHIS),  
<http://www.co.umist.ac.uk/ibhis/>
12. OWL-based Web Service Ontology,  
<http://www.daml.org/services/owl-s/>
13. RACER: Renamed ABox and Concept Expression Reasoner,  
<http://www.sts/tu-hardburg.de/~r.f.moeller/racer/index.html>
14. Semantic Grid Community Portal,  
<http://www.semanticgrid.org/>
15. Web Service Semantics – WSDL-S,  
<http://lstdis.cs.uga.edu/Projects/METEOR-S/WSDL-S/>