A FRAMEWORK FOR BIOPROFILE ANALYSIS OVER GRID

by

PIN HU

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Computing, Communications and Electronics
Faculty of Technology

January 2008

A Framework for Bioprofile Analysis over Grid

Pin Hu

Abstract

An important trend in modern medicine is towards individualisation of healthcare to tailor care to the needs of the individual. This makes it possible, for example, to personalise diagnosis and treatment to improve outcome. However, the benefits of this can only be fully realised if healthcare and ICT resources are exploited (e.g. to provide access to relevant data, analysis algorithms, knowledge and expertise). Potentially, grid can play an important role in this by allowing sharing of resources and expertise to improve the quality of care. The integration of grid and the new concept of bioprofile represents a new topic in the healthgrid for individualisation of healthcare.

A bioprofile represents a personal dynamic "fingerprint" that fuses together a person's current and past bio-history, biopatterns and prognosis. It combines not just data, but also analysis and predictions of future or likely susceptibility to disease, such as brain diseases and cancer. The creation and use of bioprofile require the support of a number of healthcare and ICT technologies and techniques, such as medical imaging and electrophysiology and related facilities, analysis tools, data storage and computation clusters. The need to share clinical data, storage and computation resources between different bioprofile centres creates not only local problems, but also global problems.

Existing ICT technologies are inappropriate for bioprofiling because of the difficulties in the use and management of heterogeneous IT resources at different bioprofile centres. Grid as an emerging resource sharing concept fulfils the needs of bioprofile in several aspects, including discovery, access, monitoring and allocation of distributed bioprofile databases, computation resources, bioprofile knowledge bases, etc. However, the challenge of how to integrate the grid and bioprofile technologies together in order to offer an advanced distributed bioprofile environment to support individualized healthcare remains.

The aim of this project is to develop a framework for one of the key meta-level bioprofile applications: bioprofile analysis over grid to support individualised healthcare. Bioprofile analysis is a critical part of bioprofiling (i.e. the creation, use and update of bioprofiles). Analysis makes it possible, for example, to extract markers from data for diagnosis and to assess individual's health status. The framework provides a basis for a "grid-based" solution to the challenge of "distributed bioprofile analysis" in bioprofiling. The main contributions of the thesis are fourfold:

A. An architecture for bioprofile analysis over grid. The design of a suitable architecture is fundamental to the development of any ICT systems. The architecture creates a

means for categorisation, determination and organisation of core grid components to support the development and use of grid for bioprofile analysis;

B. A service model for bioprofile analysis over grid. The service model proposes a service design principle, a service architecture for bioprofile analysis over grid, and a distributed EEG analysis service model. The service design principle addresses the main service design considerations behind the service model, in the aspects of usability, flexibility, extensibility, reusability, etc. The service architecture identifies the main categories of services and outlines an approach in organising services to realise certain functionalities required by distributed bioprofile analysis applications. The EEG analysis service model demonstrates the utilisation and development of services to enable bioprofile analysis over grid;

C. Two grid test-beds and a practical implementation of EEG analysis over grid. The two grid test-beds: the BIOPATTERN grid and PlymGRID are built based on existing grid middleware tools. They provide essential experimental platforms for research in bioprofiling over grid. The work here demonstrates how resources, grid middleware and services can be utilised, organised and implemented to support distributed EEG analysis for early detection of dementia. The distributed Electroencephalography (EEG) analysis environment can be used to support a variety of research activities in EEG analysis;

D. A scheme for organising multiple (heterogeneous) descriptions of individual grid entities for knowledge representation of grid. The scheme solves the compatibility and adaptability problems in managing heterogeneous descriptions (i.e. descriptions using different languages and schemas/ontologies) for collaborated representation of a grid environment in different scales. It underpins the concept of bioprofile analysis over grid in the aspect of knowledge-based global coordination between components of bioprofile analysis over grid.

Author's Declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award.

This study was funded in part by the BIOPATTERN project.

Publications:

- P. Hu, L. Sun, C. Goh, B. Hamadicharef, E.Ifeachor, I. Barbounakis, M. Zervakis, N. Nurminen, A. Varri, R. Fontanelli, S. Di Bona, D. Guerri, S. La Manna, K. Cerbioni, E. Palanca and A. Starita, "The BIOPATTERN Grid: Implementation and Applications", Proceedings of the 5th All Hands Meeting 2006, September 18-21, 2006, Nottingham, UK.
- P. Hu, L. Sun and E. Ifeachor, "An Approach to Structured Knowledge Representation of Service-oriented Grids", Proceedings of UK e-Science Programme All Hands Meeting 2007, September 10-13, 2007, Nottingham, UK.
- P. Hu, A. Anastasiou, L. Sun, E. C. Ifeachor, "A model for bioprofile over Grid in support of eHealthcare", Proceeding of the 3rd International Conference on Computational Intelligence in Medicine and Healthcare (CIMED2007), July 25-27, 2007, Plymouth, U.K.

- L. Sun, P. Hu, C. Goh, B. Hamadicharef, E. Ifeachor, I. Barbounakis, M. Zervakis, N. Nurminen, A. Varri, R. Fontanelli, S. Di Bona, D. Guerri, S. La Manna, K. Cerbioni, E. Palanca, A. Starita, "Bioprofiling over Grid for eHealthcare", Proceedings of the HealthGRID 2006 , June 7-9 2006, Valencia, Spain.
- L. Sun, P. Hu, C. Goh, B. Hamadicharef, M. Hess, E. Ifeachor, I. Barbounakis, M. Zervakis, N. Nurminen, A. Varri, "Bioprofiling over Grid for Early Detection of Dementia", Proceedings of the First International Conference on Scalable Information Systems (INFOSCALE 2006), May 30-June 1, 2006, Hong Kong.
- P. Hu, Z. Qiao, L. Sun, and E. Ifeachor, "Runtime estimation of parallel applications in computational grids", Proceedings of the 2nd International Conference on Computational Intelligence in Medicine and Healthcare (CIMED2005), June 29 - July 1, 2005, Lisbon, Portugal, pp. 472-479.
- P. Hu, L. Sun and E. Ifeachor, "A Framework for Bioprofile Analysis over Grid", Submitted to the IEEE Systems Journal.

Signed [illegible]

Date 13 Jan 2009

Word count: 42272

Acknowledgements

With a deep sense of gratitude, I wish to express my sincere thanks to my director of study Professor Emmanuel Ifeachor for his professional guidance, advices, encouragement and patience throughout this project.

I would like to thank my supervisor Dr. Lingfen Sun for her professional guidance, advices and support, and Dr. Nicolas Outram for his support during the project.

I would like to thank the BIOPATTERN project partners Dr. Ioannis Barbounakis and Mr. Nicola Pavlov from Telecommunications Systems Institute of Crete (Greece), Ms. Noora Nurminen from Tampere University OF Technology (Finland), and people from Pisa University (Italy), Sysnapsis (Greece), and Institute for the Development of New Technologies (Portugal) for their kindly collaboration in building the BIOPATTERN project grid test-bed: BIOPATTERN grid.

I would like to thank Dr. Geoff Henderson, Dr. Cindy Goh and Ms. Peng Zhao from Signal Processing and Multimedia Communication (SPMC) Research Group, University of Plymouth for their support in providing EEG data and software implementations of EEG analysis algorithms.

I would like to thank Dr. Phil Richards, Mr. Michael Hess, Mr. John Horne, Dr. Gregory Regan and Mr. Rob Douglass from Information and Communication Technology, University of Plymouth for their efforts in building the Plymouth campus grid: PlymGRID.

I would like to thank other members of SPMC: Dr Brahim Hamadicharef, Mr. Zizhi Qiao, Mr. Athanasios Anastasiou, Dr. Zhuoqun Li, Mrs. Asiya Khan, and Mr. Paul Hopkins for all the help, support and constructive discussions for this project.

I would like to thank Professor Anne Trefethen, Dr. David Wallom, Ms. Ruth Kirkham and Mr. John Pybus from Oxford e-Research Centre (OeRC) for their support during the late stage of my PhD study.

Finally, I sincerely thank my parents Yuliang Hu and Xinmei Fu, my girlfriend Ting Sun and my sister Ying Hu for their continuous encouragement and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This Chapter is organised as follows. Section 1.1 gives the research motivations behind the project. Section 1.2 addresses the research questions. Section 1.3 presents the project aim and objectives. Section 1.4 summarises the main contributions of this thesis. Section 1.5 gives the thesis outline.

1.1 Motivation

An important trend in modern medicine is towards individualisation of healthcare to tailor care to the needs of the individual. This makes it possible, for example, to personalise diagnosis and treatment to improve outcome (this is important given the problems of heterogeneity of many diseases and differences in individual patients). However, the benefits of individualisation of care can only be fully realised if healthcare and ICT resources are exploited (e.g. to provide access to relevant clinical data, and to algorithms to analyse the vast amounts of data being generated by modern medicine, and access to expertise to interpret and make use of the information derived therein). Potentially, grid can play an important role in this by allowing sharing of resources and expertise to improve the quality of care.

The integration of grid and healthcare has already formed a new exciting and specialist area called Healthgrid. Great efforts, resources and funding have been put into national, regional and international initiatives in developing healthgrid infrastructures, services and applications to support healthcare. Examples of research in the field of healthgrid include integration of distributed biomedical information for Paediatrics (e.g. the EU's Health-e-Child [4], distributed mammography data retrieval and processing (e.g. the EU's

MammoGrid [5] and the UK's eDiaMoND [6] projects), and multicentre neuro-imaging (e.g. the USA's BIRN [7] and Japan's BioGrid [8]).

The integration of grid and the new concept of bioprofile is considered as a new topic in the healthgrid community for the individualisation of healthcare. Conceptually, a bioprofile represents a personal dynamic "fingerprint" that fuses together a citizen's current and past bio-history, biopatterns and prognosis. By biopattern, we mean basic information/pattern which provides clues about the underlying clinical evidence necessary for diagnosis and treatment. It is typically derived from specific data types (e.g. genomic microarray, EEG and PET) in a single medical investigation using techniques, such as statistical signal processing, data mining and pattern recognition. Bioprofile combines not just data, but also analysis and predictions of future or likely susceptibility to disease, such as brain diseases and cancer [9].

The success of bioprofile depends on the support from Information and Communications Technologies (ICT). Bioprofile requires, for example, information from Electronic Health Records (EHR) of subjects; multimodal data (i.e. data with various data modalities, such as imaging, electrophysiology and genomics); and analysis of multimodal data in order to obtain biomarkers. Bioprofile creates not only local problems, but also global problems. This is because of the need to share clinical data between different bioprofile centres (e.g. a GP surgery and hospital) due to, for example, mobility, as a person may live in different regions so that his/her bioprofiles are distributed over different sites; insufficient and/or inadequate local storage capacity; and lack of local computation power for analysis of a large amount of data.

At a local level, most bioprofile requirements can be met by using established ICT technologies, such as, file and database management systems, which offer storage for today's EHR, multimodal data, biomarkers, etc., as well as data models, which provide functionalities to structure, organise and add semanticis to specific types of bioprofile data and domain knowledge; computational clusters (e.g. high performance and high throughput cluster), which supply computation power for analysis of bioprofile data; etc.

At a global level, established ICT technologies however do not provide solutions to the bioprofile requirements. This is mainly due to two reasons. First, different bioprofile centres are likely to be using heterogeneous IT resources. For example, for data storage and management, some sites may use MySQL [10] or PostgreSQL [11] and others may use Xindice [12], where the data schemas used in each site may also be different to each other. Because there is currently no existing, or more precisely, well-developed standards for global federation of heterogeneous resources, it is difficult to simultaneously access multiple distributed heterogeneous resources without add-on mechanisms. Second, it is impractical to access and manage distributed bioprofile resources only based on today's Internet and static resource information (e.g. IP address, port and data schemas). In theory, bioprofile should benefit every citizen's healthcare. This means that in the future there will be numerous centres, which individually hold bioprofile data that belong to a number of citizens, or computation power that can support analysis of bioprofile data. New resources may join in to support the use of bioprofile at any time. The status of each resource may also change from time to time. To realise the bioprofile concept, a set of additional techniques for resource discovery, monitoring and allocation is required in order to manage and access large amounts of bioprofile-related resources.

Grid as an emerging resource sharing technology offers possible solutions for the deployment of a persistent, standards-based service infrastructure that supports the creation of, and resource sharing within, distributed communities. It is built on top of existing ICT technologies and sharing is under highly, but decentralised control. Resources to be shared are heterogeneous in general, but inter-operated by standard, open, general-purpose protocols and interfaces [13]. Grid fulfils the needs of bioprofile in several aspects, e.g. discovery, access, monitoring and allocation of distributed bioprofile databases, computation resources, bioprofile knowledge bases, etc. However, it is not clear how to integrate the grid and bioprofile technologies together in order to offer advanced distributed bioprofile environment to support individualised healthcare.

There are many issues which can be investigated, such as security and privacy for bioprofile distribution, concurrent programming and workflows for bioprofile applications, and (distributed) bioprofile data model. It is important to research the general architecture and infrastructure, which will benefit the management and continued development of the bioprofiling over grid concept.

Bioprofile analysis is a critical part of bioprofiling. It concerns how to carry out analysis on individual or a set of bioprofiles in order to refine abstractions for either clinical investigation (e.g. diagnosis support) or research activities (e.g. the investigation of analysis algorithms and fusion of biopatterns). The application usually requires the support of bioprofile retrieval, on demand analysis computation and analysis-related knowledge provisioning.

To provide a basis for bioprofile analysis over grid, it is necessary to investigate 1) major issues that are involved in delivering secure, reliable, seamless and efficient grid environment for bioprofile analysis; 2) architecture for bioprofile analysis over grid; 3) service model for classification and clarification of services to support various distributed bioprofile analysis applications; 4) demonstration of grid environment for bioprofile analysis; and 5) methods of enhancing coordination between components of bioprofile analysis over grid.

There are several research hypotheses behind these investigations. First, if grid offers solutions for the deployment of a persistent, standards-based service infrastructure that supports the creation of, and heterogeneous resource sharing, within distributed communities, then it will provide solutions for distributed bioprofile analysis. Second, the design of a bioprofile analysis over grid architecture can contribute to the categorisation, determination and organisation of core bioprofile analysis over grid components. Third, if some existing services and/or tools provide generic grid functionalities, then these services and/or tools can be directly utilised or be organised to support distributed bioprofile analysis. Fourth, if the success of a grid service is related to the delivery of usability, flexibility, extensibility and reusability, then the issues of "application-driven development", "separation of concerns", "layered services" and "generic" will be important in the successful design of a grid service. Fifth, if it is impractical to use a single ontological description to describe an entire grid environment, then a group of organised multiple (heterogeneous) descriptions can be doable.

1.2 Research Questions

This thesis is intended to address the following questions:

A. What are the main components of bioprofile analysis over grid and what should their relationships be?

This is a fundamental question in the building of an environment for bioprofile analysis over grid. It should provide an understanding of the general concept of bioprofile analysis over grid, classification and clarification of main components of bioprofile analysis over grid, and a basis for the long term extensible and collaborative development of a distributed environment for bioprofile analysis in support of individualised healthcare. The investigation will focus on:

(a) Determination of bioprofile characteristics and bioprofile analysis requirements.

(b) Identification of what the basic elements of bioprofile analysis over grid should be.

(c) How to build grid environment for bioprofile analysis?

(d) Determination of what the architecture of bioprofile analysis over grid should be.

Detailed discussion can be found in Chapter 3.

B. How to develop and manage services to support distributed bioprofile analysis applications?

This is a further investigation based on the proposed architecture of bioprofile analysis over grid. Service is a core grid component. Normally, it provides platform-independent protocols and standards used for exchanging data between clients and resources. A service is developed to provide functions (e.g. query into relational databases) required by one or more applications. This investigation should provide the knowledge in construction and organisation of services to support distributed bioprofile analysis applications. The main issues that needed to be addressed include:

(a) How to design services to meet the requirements of collaboration, extensibility and reusability?

(b) What should be the service architecture for bioprofile analysis over grid?

(c) How to utilise existing services and how to develop new services to meet requirements defined by bioprofile analysis applications?

Detailed discussion can be found in Chapter 4.

C. How to develop a grid-enabled bioprofile analysis environment to demonstrate the bioprofile analysis over grid concept?

This development should lead to three deliverables: 1) Grid test-beds, which will provide distributed platforms for current and future research in bioprofiling over grid; 2) A guidance in the implementation and integration of new (grid) mechanisms with existing bioprofile analysis and ICT related techniques together to enable bioprofile analysis over grid; 3) A grid-enabled bioprofile analysis environment, which can be used to support biomedical research activities, clinical investigation, etc. The development will focus on the following:

(a) How to build grid test-bed(s) in order to provide distributed platform(s) for bioprofiling over grid research?

(b) What mechanisms can be utilised and/or should be implemented in order to deliver the required distributed bioprofile analysis functionalities

(c) How existing and implemented mechanisms can be organised to provide collaborated functions to realise bioprofile analysis over grid?

(d) What can be achieved and what is limited in the implementation?

Detailed discussion can be found in Chapter 5.

D. How to deliver knowledge support to strengthen global coordination between components of bioprofile analysis over grid?

   This is important in knowledge-based global coordination between different components of bioprofile analysis over grid (e.g. resources, applications, security, quality of service and users). However, hitherto only limited work has been done in utilising and organising various knowledge representation languages and schemas/ontologies to provide collaborated knowledge representation of grids. Addressing this research question should lead to a knowledge representation scheme, which can offer a flexible and extensible way to organise multiple (heterogeneous) descriptions of individual grid entities together for knowledge representation of grid environments. The scheme should benefit not only bioprofile analysis over grid, but also other kinds of grid environments. Main issues that need to be addressed include:

   (a) What major entities need to be considered in representing the bioprofile analysis over grid concept?

   (b) What should the architecture be for knowledge representation of grid-enabled bioprofile analysis based on existing knowledge representation techniques?

   (c) How to design and implement mechanisms to support such knowledge representation?

   Detailed discussion can be found in Chapter 6.

1.3 Research Aim and Objectives

The aim of this project is to develop a framework for bioprofile analysis over grid to support individualised healthcare. Bioprofile analysis is a critical part of bioprofiling. Analysis makes it possible, for example, to extract markers from data for diagnosis and to assess individual's health status. The framework provides a basis for a "grid-based" solution to the challenge of "distributed bioprofile analysis" in bioprofiling. Specific objectives of the research are to:

A. Propose an architecture for bioprofile analysis over grid that determines main system/environment components, component relationships and overall structure of bioprofile analysis over grid;

B. Develop a service model for bioprofile analysis over grid. The service model should address service design principles, deliver a distributed bioprofile analysis service architecture and provide an example of the approach that theoretically demonstrates the utilisation and development of services to enable bioprofile analysis over grid;

C. Develop a grid-enabled bioprofile analysis environment. This involves the development of grid test-bed(s) to support long-term research in bioprofiling over grid. This should also contribute to knowledge in the implementation and integration of new (grid) mechanisms with existing bioprofile analysis and ICT related techniques together for enabling of bioprofile analysis over grid;

D. Develop a knowledge representation scheme to semantically strengthen the global coordination between different components of bioprofile analysis over grid. The scheme should target the knowledge representation of the entire grid environment, and focus on the utilisation and organization of various knowledge representation languages and schemas/ontologies. The scheme should be generic, so applicable to other sorts of grid environments.

1.4 Contributions

The contributions of this project are based on existing grid middleware tools, ICT, bioprofile analysis, knowledge representation, and other associated technologies. The main contributions are:

A. An architecture for bioprofile analysis over grid. The design of a suitable architecture is fundamental to the development of any ICT systems. The architecture creates a means for categorisation, determination and organisation of core grid components to support the development and use of grid for bioprofile analysis;

   (The associated publications are [14] [15] [16], and the paper submitted for the publication is [17])

B. A service model for bioprofile analysis over grid. The service model proposes a service design principle, a service architecture for bioprofile analysis over grid, and a distributed EEG analysis service model. The service design principle addresses the main service design considerations behind the service model, in the aspects of usability, flexibility, extensibility, reusability, etc. The service architecture identifies main categories of services and outlines an approach in organising services to realise certain functionalities required by distributed bioprofile analysis applications. The EEG analysis service model demonstrates the utilisation and development of services to enable bioprofile analysis over grid;

(The paper submitted for the publication is [17])

C. Two grid test-beds and a practical implementation of EEG analysis over grid. The two grid test-beds: the BIOPATTERN grid and PlymGRID are built based on existing grid middleware tools as well as other ICT technologies. They provide essential experimental platforms for research in bioprofiling over grid. The practical implementation demonstrates how resources, grid middleware and services can be utilised, organised and implemented to support distributed EEG analysis for early detection of dementia. The developed distributed EEG analysis environment can be used to support a variety of research activities in EEG analysis;

(The associated publications are [14] [15] [16], and the paper submitted for the publication is [17])

D. A scheme for organising multiple (heterogeneous) descriptions of individual grid entities for knowledge representation of grid. The scheme can solve the compatibility and adaptability problems in managing heterogeneous descriptions (i.e. descriptions using different languages and schemas/ontologies) for collaborated representation of a grid environment in different scales. It underpins the concept of bioprofile analysis over grid in the aspect of knowledge-based global coordination between components of bioprofile analysis over grid.

(The associated publication are [18] [19])

Since this work is a part of the BIOPATTERN project, it is important to note that the contribution C: "two grid test-beds and a practical implementation of EEG analysis over grid" is a collaborative contribution. The author of this thesis: Pin Hu was responsible for building and testing the Plymouth part of the BIOPATTERN grid test-bed, and the backend part of PlymGRID. The author also developed the BIOPATTERN grid portal and most of the bioprofile analysis-related services. Other project members have contributed to the building and testing of non-Plymouth grid nodes, the EEG data and software implementations of EEG analysis algorithms, as cited in the acknowledgements of this thesis.

1.5 Thesis Outline

The outline of the thesis is described as follows:

- Chapter 2 gives brief background information about the state-of-the-art development in the fields of grid, healthgrid and knowledge representation;
- Chapter 3 presents the general concept of bioprofile and bioprofile analysis, overviews the framework for bioprofile analysis over grid, and proposes an architecture for bioprofile analysis over grid;
- Chapter 4 proposes a service model for bioprofile analysis over grid. Within this chapter, a service design principle is discussed. A service architecture for bioprofile analysis over grid is proposed. An EEG analysis service model is illustrated;
- Chapter 5 presents the implementation of a grid environment in support to EEG analysis. Within this Chapter, an EEG analysis scenario is introduced. Two grid test-bed(s) are presented. Details of utilisation, implementation and organisation of individual EEG analysis, grid and other ICT mechanisms to enable EEG analysis over grid are given. The achievements and limitations of the implementation are discussed;
- Chapter 6 proposes a knowledge representation scheme which offers a way to organise multiple (heterogeneous) descriptions of individual grid entities together for knowledge representation of grid. Within this Chapter, an extended model for bioprofile analysis over grid is presented. A schematic multiplex description architecture for organising heterogeneous descriptions of knowledge representation of grid are proposed. The design of a kernel ontology skeleton (as a part of the description architecture) is discussed. Associated implementations are presented;

- Chapter 7 concludes the thesis and includes discussions of the main contributions of this work, and the limitations and possible future work.

Chapter 2

Background

2.1 Introduction

This chapter aims to give a background and an overview of the main technologies involved in this project. It is organised as follows. Section 2.2 reviews early approachs in the use of middleware for the development of distributed biomedical systems. Section 2.3 reviews the up-to-date development of grid technology, including the current de facto grid architecture, infrastructure and middleware tools. Section 2.4 overviews the healthgrid concept. Section 2.5 introduces the knowledge representation techniques and the Semantic grid concept. Section 2.6 discusses the related work. Section 2.7 concludes this chapter.

2.2 Middleware and Distributed Biomedical System

Middleware is a common IT term today, which could be described as any computer software that allows other software to interact [20]. It is mostly designed to help manage the complexity and heterogeneity inherent in distributed systems [21]. It can be divided into different categories, including message oriented middleware, object middleware, RPC middleware, database middleware, transaction middleware and portals [20].

Between 5 and 15 years ago, the most popular type of middleware used to develop distributed biomedical systems was the object middleware, which provided the abstraction of an object that is remote yet whose methods could be invoked just like those of an object in the same address space as the callers [21]. The studies of such systems can be found in many research documents, such as distributed processing of large biomedical 3D images [22], telemedicine system for medical image analysis and modelling [23], and bioinformatics on java distributed system [24].

Java Remote Method Invocation (Java RMI) [25] and Common Object Requesting Broker Architecture (CORBA) [26] are two major standards of object middleware used in many object middleware-based distributed biomedical systems. The former is a Java application programming interface for performing the object equivalent of remote procedure calls. The latter is a standard defined by the Object Management Group (OMG) [26] that enables software components written in multiple computer languages and run on multiple computers to working together.

Both Java RMI and CORBA use optimised connection-oriented communications protocols that are either language specific, or have detailed rules defining how data-structures and interfaces should be realised. This implies that the communicating objects are tightly coupled. The feature may deliver efficient communications, but has less flexibility, scalability, replaceability and fault tolerance, as compared with the Service Oriented Architecture (SOA) [27]. This use of object middleware is therefore not suitable for large and complex distributed biomedical systems or environments.

2.3 Grid

The term "Grid" was coined in the mid 1990s to denote a proposed distributed computing infrastructure for advanced science and engineering [28]. The common definition of grid can be found in [29], as "sharing environments implemented via the deployment of a persistent, standards-based service infrastructure that supports the creation of, and resource sharing within, distributed communities". The notion of resource sharing may include direct access to shared local or global computers, software, data, and other resources (e.g. earthquake shake tables and x-ray research facilities). The sharing is under high, but decentralised control. Resources to be shared are heterogeneous in general, but inter-operated by standard, open, general-purpose protocols and interfaces [13].

With grid, organisations can optimise computing, data and other resources, pool them for large capacity workload, share them across networks, and enable collaboration. It provides seamless and interactive resource sharing environments. It offers large and remote software operations. It allows users to access distributed data in a convenient and effective manner. It also utilises idled and distributed computation resources to offset the shortage of computing power for large amounts of computation-intensive applications with a flexible and efficient form.

Grid can support applications in a number of scientific and business fields, such as biomedicine, geography, astrography, finance and ecommerce. In scientific area, grid-based applications may include distributed computing (e.g. scientific simulation), high-throughput applications (e.g. RSA keycracking), and data-intensive applications (e.g. digital library). In business area, grid-enabled applications may include information sharing and trading.

Grid research has been widely spread all over the world. EGEE [30], CoreGRID [31], and NextGRID [32] are just few examples of grid projects. In grid projects, certain key problems of grids have been studied, which are grid flexibility, security, efficiency, resource collaborativity and connectivity, resource and service transparency, service availability, etc.

2.3.1 Open Grid Services Architecture

Open Grid Services Architecture (OGSA) [1] is a well-known service-oriented architecture, which defines a set of core capabilities and behaviours that address key concerns in grids (e.g. security, resource/service discovery and management of virtual organisations). It aims to facilitate the seamless use and management of distributed, heterogeneous resources by a set of capabilities.

Figure 2.1 presents some of the capabilities concerned by OGSA. As can be seen, OGSA classifies these capabilities into three major logical and abstract tiers, as base resources, (higher level) virtualisation/abstraction and applications. It realises the logical middle layer in terms of services, the interfaces these services expose, the individual and collective state of resources belonging to these services, and the interaction between these services within a service-oriented architecture. In OGSA, services are divided into 7 major categorises, as:

- Infrastructure services, which provide common components (e.g. naming, representing state and notification) for service build-up;
- Execution management services for instantiating and managing, to completion, units of work (i.e. Finding execution candidate locations; Selecting execution location; Preparing for execution; Initiating the execution; Managing the execution);
- Data services for movement, access and update of data resources. They may also provide the capabilities necessary to manage the metadata that describes OGSA data services or other data, in particular the provenance of the data itself;

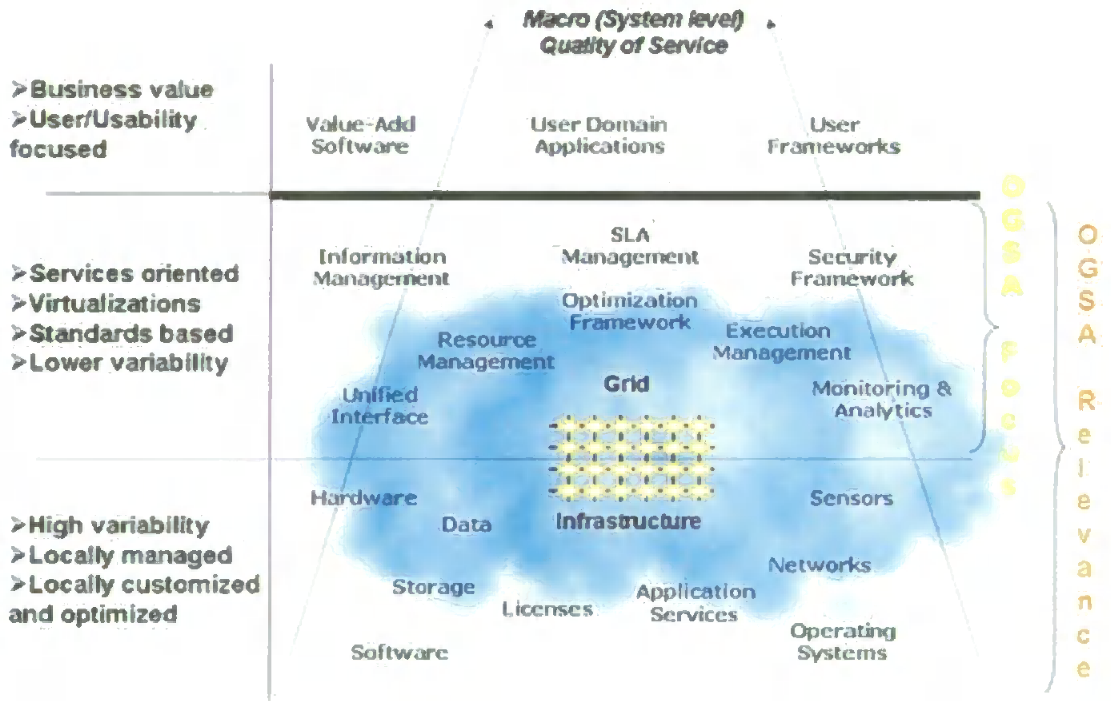


Figure 2.1: OGSA concerned capabilities for the use and management of distributed, heterogeneous resources [1]

- Resource management services, which perform several forms of management on resources in a grid, including management of the resources themselves, management of the resources on grid, and management of the OGSA infrastructure;
- Security services to facilitate the enforcement of the security-related policy within a virtual organisation;
- Self-management services, which provide several forms of self-management mechanisms (e.g. self-configuring, self-healing and self-optimising) in helping reduce the cost and complexity of owning and operating an IT infrastructure;
- Information services, which offer the ability to efficiently access and manipulate information about applications, resources and services in grids.

OGSA does not address issues of programming models, programming languages, implementation tools, or execution environments. All these tasks are given to hosting environments to deal with. In concept hosting environments provide either low levels of functionality (e.g. native operating system processes), or superior functionalities (e.g. programmability, manageability, flexibility and safety). Host environments with superior functionalities are normal container or component based. Examples of container/component-based hosting environments are J2EE [33], WebSphere [34], and .NET [35].

2.3.2 Web Service Resource Framework

Web Service Resource Framework (WSRF) [36] is a generic framework for modelling and accessing persistent resources use Web services to ease service definition, implementation, integration and management. It provides a solid infrastructure, which supports the compostion of OGSA. WSRF is the replacement of Open Grid Services Infrastructure (OGSI) [37] due to some criticisms on OGSI addressed by the Web service community (e.g. too much in one specification; too object oriented; and incompatible with Web Service Definition Language (WSDL) 1.1).

WSRF is concerned primarily with the creation, addressing, inspection, and lifetime management of stateful resources. WSRF provides the means to express state as stateful resources and codifies the relationship between Web services and stateful resources in terms of the implied resource pattern, which is a set of conventions on Web services technologies, particularly XML, WSDL, and WS-Addressing. WSRF is defined by five normative specifications, as

- WS-ResourceLifetime, which is a set of mechanisms for WS-Resource destruction, including both time based (i.e. scheduled destruction) and message exchange based destroying or termination of resources;
- WS-ResourceProperties, which defines type and value of WS-Resource state, and includes mechanisms for retrieving, changing, and deleting WS-Resource properties;
- WS-RenewableReferences, which consists of mechanisms for retrieving and updating an endpoint when it becomes invalid;
- WS-ServiceGroup, which is an interface for representing and managing heterogeneous by-reference collections of Web services;
- WS-BaseFaults, which is a base fault XML type for use when returning faults in a Web services message exchange;

- WS-Notification, in addition to WSRF, defines a general, topic based Web service system for publish and subscribe interactions that build on WSRF.

2.3.3 Grid Middleware Tools

A. Globus Toolkit 4

Globus Toolkit (GT) is a software package, which contains a set of middleware components to support the development of service-oriented distributed computing applications and infrastructures [38]. It has been developed since the late 1990s and has been delivered in three major versions, as pre-Web services based GT2, OGSI-enabled GT3 and WSRF-enabled GT4.

GT4, as the latest version of Globus Toolkit to date, makes extensive use of Web services mechanisms to define its interface and to structure its components. It is a realisation of the OGSA requirements and a sort of de facto standard for the Grid community. GT4 implements a set of (infrastructure) services on top of WSRF [39]. These services address most concerns identified in OGSA, including execution management, data access and movement, replica management, monitoring and discovery, credential management, and instrument management [38]. Figure 2.2 presents the relationship between Web services, OGSA, WSRF and GT4.

GT4 also contains three major containers for hosting user-developed services written in Java, Python and C, respectively. These containers provide implementation of security management, discovery, state management, and other mechanisms frequently

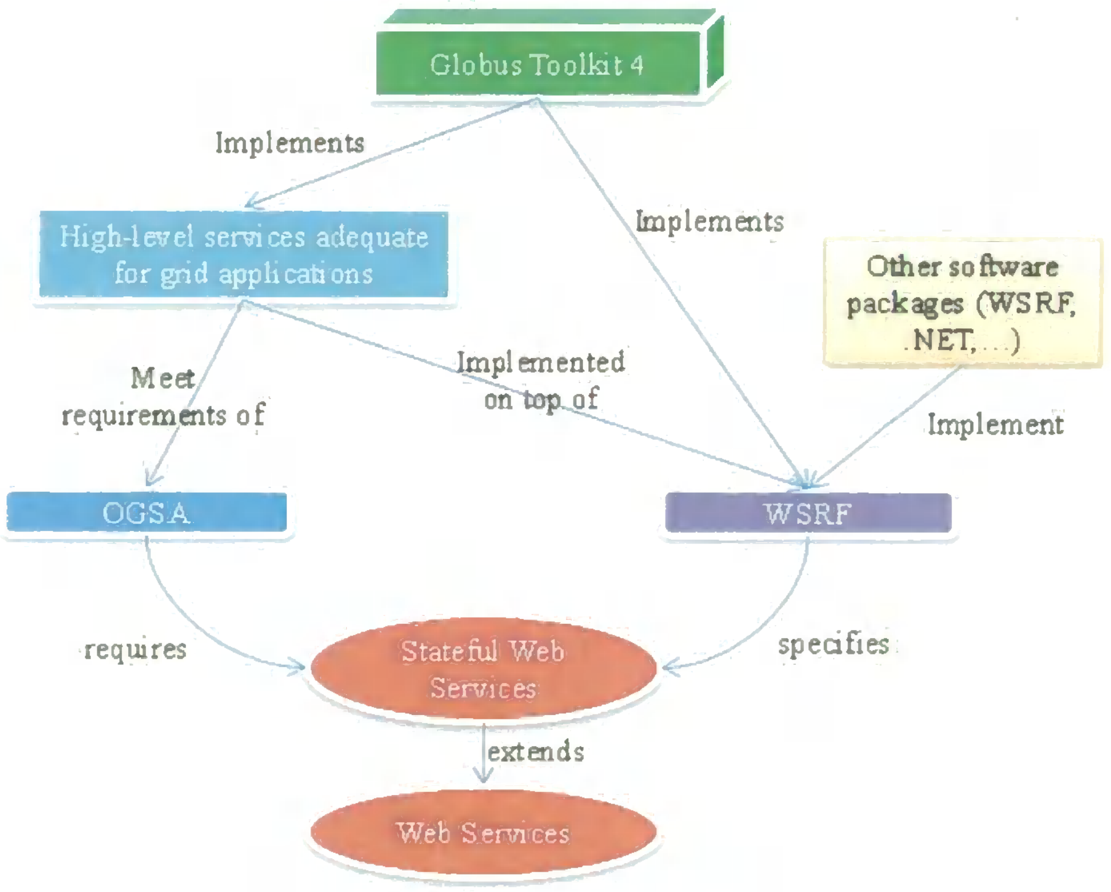


Figure 2.2: Relationship between Web services, OGSA, WSRF and GT4

required when building services. GT4 client libraries are provided to allow client programs to invoke operations on both GT4 and user-developed services [38].

B. OGSADAI

The Open Grid Services Architecture - Data Access and Integration (OGSADAI) is a middleware product which allows data resources (e.g. relational and XML databases) to be exposed on to grids. It offers various interfaces to support many popular database management systems (e.g. MySQL [10], PostgreSQL [11] and Oracle [40]). Additional functions OGSADAI can provide include querying, transforming and delivering data over distributed data environments [41].

OGSADAI has three major products, as OGSADAI-WSRF, which is compatible with the GT implementation of WSRF; OGSADAI-WSI, which is compatible with the UK OMII's implementation of Web Services Inter-operability (WS-I) [42]; and OGSADQP, which supports distributed queries over OGSADAI data services.

C. Other Tools

Besides GT4 and OGSADAI, today, there are many other grid middleware tools. For instance, Storage Resource Broker (SRB) [43] for support of sharing of distributed and heterogeneous storage systems; UNICORE [44], which contains both client and server software to make workflow-based access of distributed data and computing resources; gLite [45], which integrates a set of grid middleware tools (including GT series) to make easy building of grid applications.

2.4 Healthgrid

2.4.1 overview

Healthgrid is the application of grid to healthcare. Today, a common definition of healthgrid is stated as, *"A healthgrid is an environment in which data of medical interest can be stored and made easily available to different actors in healthcare systems, such as physicians, healthcare centres, patients and citizens in general"* [46] [47]. In the definition, the notion of data of medical interest may include five levels of biomedical information, as public health, patient, tissue and organ, cell, and molecule. The sharing of such data must be guaranteed in three aspects, as security, respect for ethics, observance of regulations [46].

The definition has covered most, but not all areas of healthgrid. There are many new concepts of healthcare (e.g. bioprofile) coming out. Those new concepts are leading to the healthgrid focusing not just on data sharing, but also distributed health data analysis and other means of biomedical investigation, which are inseparable to the sharing of distributed resources. One example of distributed health data analysis is the one we will discuss in this thesis, as bioprofile analysis.

Healthgrid has many applications in biomedicine, bioinformatics, medical informatics, primary/acute healthcare and social services. It can be applied to improve individualised healthcare and to support studies, such as dementia, cancer and epidemiology. For individualised healthcare, healthgrid can facilitate the access of relevant health information of a patient regardless of where he/she has been to or where he/she is now; the use of the computer-aided tools for the specific data of the patient for the detection and diagnosis of a disease to support physician's decision-making; and the definition of the most suitable therapy and treatment from the diagnosis.

The development of healthgrid is still in a very early stage [47] [48]. Mark, et al in "SHARE Roadmap 1" [47] suggested two steps of technical road map in the aspects of the development of computing, data and knowledge grid nodes in medical research centres, and the production of a standard for the exchange of medical images as well as EHR on grids. Other healthgrid research activities carried out by other projects include, for example, integration of distributed biomedical information for Paediatrics (e.g. the EU's Health-e-Child [4], distributed mammography data retrieval and processing (e.g. the EU's MammoGrid [5] and the UK's eDiaMoND [6] projects), and multicentre neuro-imaging (e.g. the USA's BIRN [7] and Japan's BioGrid [8]).The obtained outcomes of these researches, to date, are mainly healthgrid prototypes and tools for specific healthcare applications acquired knowledge and experience in the delivery of healthgrids.

2.4.2 Healthgrid Requirements

Healthgrid is grid. It has many general requirements, as a normal grid does.

A. Data, computation and knowledge related.

Healthgrid needs to provide access to distributed biomedical data, computational resources and knowledge bases. The provisioning should be transparent to users so that users do not need to have knowledge about aspects, such as resource locations and how to use and integrate them.

B. Networking.

Healthgrid requires reliable and efficient data transmission between distributed resources. This is essential to, for example, access integrated and distributed analysis of biomedical data. The requirements of transferring a set of data (e.g. bandwidth and latency) and actual size of data are varied with different applications (e.g. MRI analysis and access of personal information).

Grid is not healthgrid, as healthgrid has specific requirements compared with a normal grid.

C. Security

Healthgrid has more rigorous security requirements compare to other types of grid (e.g. grid for high energy physics and geography). In general, healthgrid needs three essential types of security enforcements, as authentication, authorization and auditing. The nature of high sensitivity of biomedical information and personal data leads to more stringent secure mechanisms to be enforced in data access, transmission, storage and processing.

D. Privacy and confidentiality

Personal biomedical data is confidential. In order to keep the privacy and protect confidentiality of patients, biomedical data need to be anonymised (remove sensitive patient data from file header), deidentified (e.g. face de-identification for a structural

MRI scan of a head), or encrypted to guarantee its confidentiality and integrity.

E. Legal and ethical issues

The use of anonymised information needs to satisfy the requirements under the Data Protection Act as well as gaining clearance from local ethics committee for research involving just a local site, or to a multi-site ethics committee for clearance to use data across many sites. The use of data normally needs explicit consent from the patients.

F. Quality of Service

Quality of Service (QoS) requirements (e.g. required response time, required audio/video/image quality) differ with biomedical applications. For example, remote surgery requires real-time high quality audio/video quality transmission. Obstetrics baby monitoring during labour also requires near-real-time response time if grid is involved in case-based decision support.

2.4.3 Electronic Health Record and Healthgrid

Electronic Health Record (EHR) is a term referring to an individual's medical record information in digital format [49] [50]. The notion of information includes patient medical record (as can be found in today's EHR systems), multimodal data (e.g. electrophysiology data and medical images), and biomarkers. It can aid clinicians' decision-making by providing access to patient health record information where and when they need it and by incorporating evidence-based decision support. It plays many important roles in healthcare, such as representation of a provider-based view of that patient's health history; a method for clinical communication and care planning among the individual healthcare practitioners serving the patient; a source of data for clinical, health services, outcomes research, and public health; and a major resource for healthcare practitioner education [49].

EHR systems and associated standards and tools are being developed worldwide. In the UK, the National Health Service (NHS) [51] has been planning since 1998 to develop a nationwide Integrated Care Record Service (ICRS) providing health care providers and patients with 24 hour on-line access to EHRs on a central data-spine [52] [53]. EHR related projects, such as HL7 [54] and OpenEHR [55], have also been contributing to EHR standards, open source software and tools in the technical space for EHR systems.

With the development of the EHR technology, the interoperability between distributed EHR systems has already been focused on by some healthgrid projects, such as the aforementioned SHARE project [47] and the DELOS project [56]. The notion of interoperability may include access to distributed EHR systems, and the transformation as well as the integration of EHRs. The requirements of healthgrid applying to EHR are, in general, similar to those requirements introduced above, but are even more stringent in the aspects of security, privacy and confidentiality since most EHRs involve large amounts of personal information.

2.5 Knowledge Representation

2.5.1 Overview

Knowledge representation is a term used in both cognitive science and artificial intelligence. Its notion can be understood in terms of five distinct roles it plays, as a surrogate; a set of ontological commitments; a fragmentary theory of intelligent reasoning; a medium for pragmatically efficient computation; and a medium of human expression [57].

There are many techniques, which can be used to represent knowledge with specific interests. Examples are [57][58][59]:

- Controlled vocabulary, which is a carefully selected list of terms for the tag of units of information so that each term describes only one concept and vice versa;
- Taxonomy, which is a collection of controlled vocabulary terms, typically related by parent-child relationships and so organised into a hierarchical structure;
- Thesaurus, which uses associative relationships in addition to parent-child relationships for networking controlled vocabulary terms in a given domain of knowledge;
- Faceted classification, which allows the assignment of multiple classifications to an object, enabling the classifications to be ordered in multiple ways;
- Ontology, which is a term being used in many different ways, such as a glossary, data dictionary, thesaurus, taxonomy, schema, and a data model. In computer science,

an ontology may be described as a model that represents a set of concepts within a domain and the relationships between those concepts.

Knowledge can be presented in various ways, expressed using various languages, and stored in various mediums. Main knowledge representation formalisms include [60][61]:

- Semantic networks, which are graphs where each graph contains nodes to represent concepts and arcs/slots to represent relations between these concepts;
- Frames and scripts, which have similar structures for representing stereotypic knowledge and expectations which would allow a system to impose coherence on incoming information;
- Production (rule) systems, which allow for the simple and natural expression of "if-then" rules;
- Logic, which employs the notions of constants, variables, functions, predicates, logical connectives and quantifiers in order to represent known facts and deduce new facts in logical formulas; etc.

There are many languages, which can be used for knowledge representation. For example, in simple cases, knowledge may be represented as a set of related tables and stored in relational databases; or described and held in plain XMLs [62]. In the Semantic Web society, several languages for knowledge representation have been developed based on HTML (e.g. SHOE [63]) and XML (e.g. RDF [64] and OWL [65]).

The Resource Description Framework (RDF) is a language recommended by the W3C [66] standardisation body for representing information in the Web. It was originally designed as a metadata model but it has come to be used as a general method of modelling information. RDF can be used to represent information about just anything. It provides a general, flexible method to decompose knowledge into small pieces, called triples (i.e. subject-predicate-object), with some rules about the semantics of those pieces. RDF's simple data model and ability to model disparate, abstract concepts has also led to its increasing use in knowledge management applications unrelated to Semantic Web activity. RDF Schema (RDFS) in addition to RDF further facilitates the specification of application-specific ontological vocabularies in the form of class and property hierarchies on top of RDF resources.

The Web Ontology Language (OWL) is another W3C recommended language for knowledge representation. It is designed for use by applications that need to process the information content instead of just presenting information to humans. OWL provides additional vocabulary along with a formal semantics to facilitate most advanced machine interpretability of Web content than that supported by XML and RDF(S). It comes with three different flavours, namely OWL-Lite, OWL-DL and OWL-Full, reflecting different degrees of expressiveness. OWL-Lite and OWL-DL are Description Logic [67] like products to support users who primarily need a classification hierarchy and simple constraints and who want the maximum expressiveness while retaining computational completeness, respectively. OWL-Full is designed to provide compatibility with RDF(S), in order words, for users who want maximum expressiveness and syntactic freedom of RDF with no computational guarantees.

2.5.2 Semantic Web/Grid

The Semantic Web is an evolving extension of the current Web in which Web content can be expressed not only in natural language, but also in a format that can be read and used by machines [68]. It makes the automation of finding, sharing and integrating information simpler. Its development has already had great contributions to both Web and artificial intelligence communities, such as techniques for the merging of knowledge from different sources, and advanced languages (e.g. RDF and OWL) for knowledge representation.

The Semantic Grid is known as an important part of grid technology, which deals with an extension of the current grid where information and services are given well-defined meaning through the use of machine-processable descriptions to maximize the potential for sharing and reuse [69]. It utilises existing knowledge representation techniques (i.e. language, and tools for query and reasoning). The current vision of the semantic grid can be considered as the application of Semantic Web technologies both on and in grid due to the trend in using the service-oriented architecture [70].

In current thinking [69], a semantic grid normally concerns issues in 12 aspects, including:

- Resource description, discovery and use;
- Process description and enactment;
- Autonomic behaviour;
- Security and trust;
- Annotation;

- Information integration;
- Synchronous information streams and fusion;
- Context-aware decision support;
- Communities;
- Smart environments;
- Integration with legacy IT systems.

2.6 Related Work

2.6.1 HealthGrid projects

A. MammoGrid

MammoGrid [5] is a project that aims to deliver a set of evolutionary prototypes to demonstrate that "mammogram analysts" (i.e. specialist radiologist working in breast cancer screening) can use a grid information infrastructure to resolve common image analysis problems.

The project concentrated on the delivery of a set of services that addresses user requirements for distributed and collaborative mammogram analysis. The prototype P1 of the project enables the simple query and retrieval of mammograms from files distributed across different databases via "Grid-boxes" (i.e. grid nodes). The prototype P2 offers further services to enable the information exchange between the

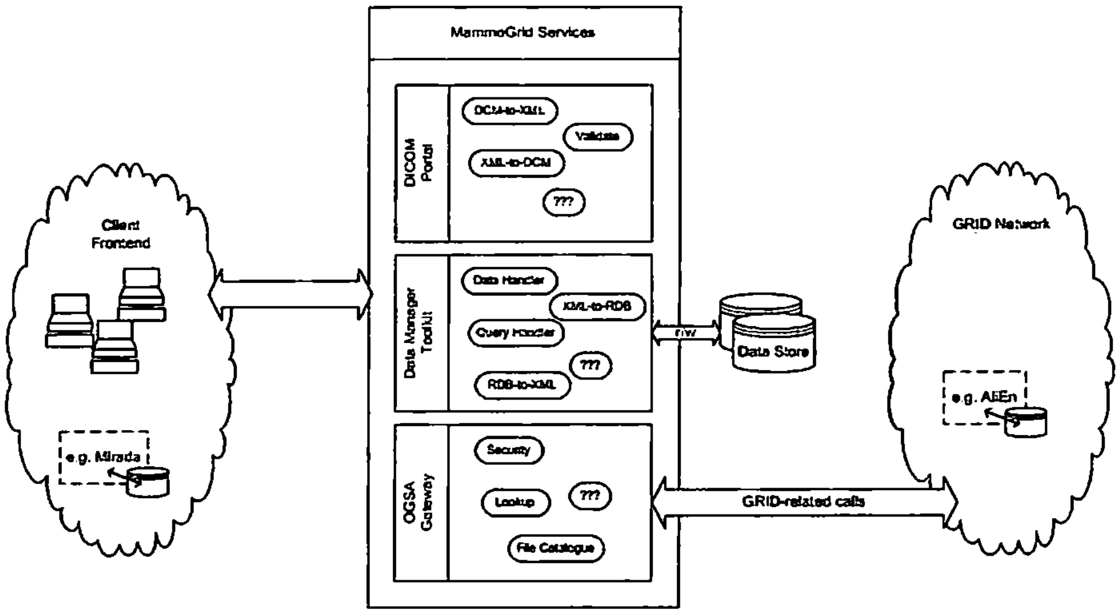


Figure 2.3: The MammoGrid Prototype 2 (P2) architecture [2]

clinician's workstation and the Grid-boxes, and a more loose coupling between services as compared with the prototype P1 [2].

The MammoGrid prototype P2 was built based on a Service-Oriented Architecture with portals/gateway/interfaces both to external systems (e.g. the MammoGrid image acquisition hardware) and to grid. Figure 2.3 shows the architecture of the prototype P2. The architecture comprises a set of MammoGrid services, where the DICOM Portal facilitates information exchange, application-level translation and validation; the Data Manager Toolkit enables queries to data stores; and the OGSA Gateway delivers security, look-up and file-catalogue services to grid networks.

The MammoGrid project provides a very good foundation to the development of healthgrid. It addresses certain healthgrid requirements (i.e. data, networking and security) and possible solutions to those requirements. The scope of the project is limited to the store, query and retrieval of images. The architecture and implementation of MammoGrid is therefore specific to imaging applications, and may not be applied to other healthcare applications.

B. BIRN

Biomedical Informatics Research Network (BIRN) [7] is a project fostering distributed collaborations in biomedical science by utilising information technology innovations. It aims to address the needs of biomedical researchers to access, exchange and analyse a variety of data from different research groups [3].

The project mainly focuses on brain research involving neuro-imaging to take advantage of the highly advanced level of sophistication of this community. There are three main components that deal with image and volume data in BIRN [3]:

- "Raw" volume and surface data (e.g. data that come out of the MRI scanner) which are made available to all participating researchers;
- Analysis programs which are made available to all participating institutions;
- Raw data and the results of analysis which are stored in distributed databases that can be queried by all researchers.

The BIRN project proposed a system, as presented in Figure 2.4, to facilitate functionalities which fulfil those needs in dealing with image and volume data [3].

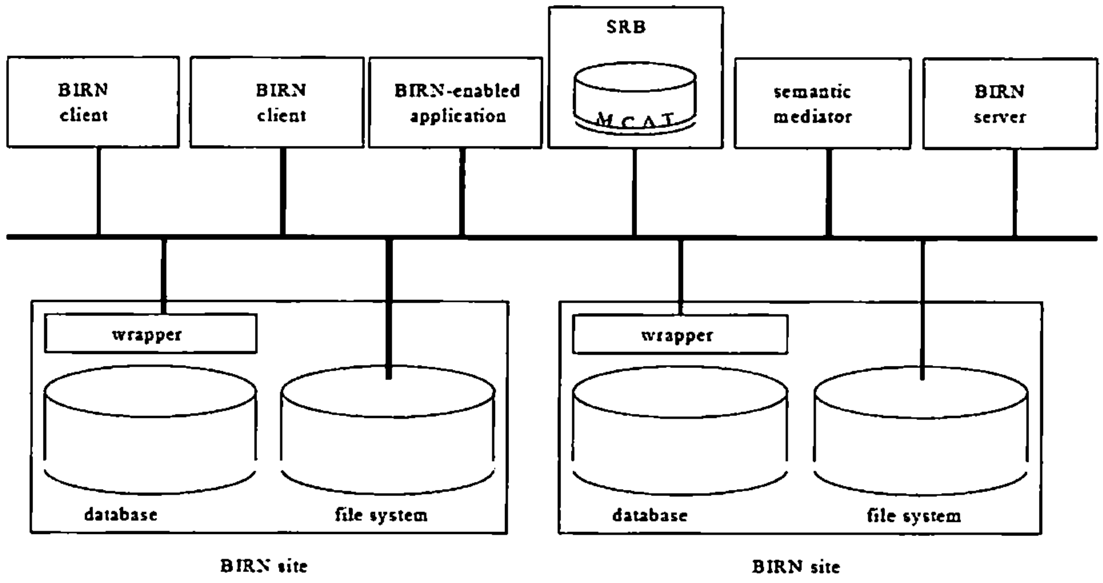


Figure 2.4: Structural diagram of the BIRN system [3]

The system consists of three main access pathways: the first gives fast access to the data of all the participating groups; the second allows the use of remote applications to process data; and the third allows searches of the databases across different groups. The system uses Storage Resource Broker (SRB) [43] to manage all remote data, a workflow engine to deliver data to be processed to applications and to return results of processing, and a mediator to collect data schema information from the various databases and integrate them into single unified view.

The BIRN project initiates the development of a distributed system to universally support biomedical research (i.e. management, upload, organisation, discovery, access, analysis and publication of biomedical and research data). The system is however "tightly coupled", when the scale of the system grows, it will be difficult to manage and coordinate distributed resources. It is thus necessary to introduce the latest grid concept in order to improve the system.

2.6.2 Knowledge Representation of Grid

Knowledge representation of grid is important in delivering semantics to grid mechanisms for automating information exchange, sharing and integration between grid entities. It is not an easy task since the concept of grid covers many computing and communication areas, such as virtualisation, data storage and management, distributed computation, networking and security.

A number of studies related to knowledge representation of grid have been carried out over the past five years. Most of the studies have focused on knowledge representation in some specific sub-domains of grid, such as grid resources and services. However, there is less research with regard to the description of multiple parts of, or even an entire, grid environment that we believe is significant in knowledge-based global coordination between different grid-related components, such as resources, applications, security, quality of services and users.

Semantic matching for grid resources [71], OntoGrid [72], Earth System Grid Semantics [73], Semantic Grid Service Discovery [74], Semantic Grid Services [75] and ontologies for Grid Service Discovery [76] are examples of the efforts which have been made on the knowledge representations of grid sub-domains to support different grid-related purposes, such as the discovery and matching of grid services and resources. Most of the outputs of those researches are ontologies and/or schemas. They can potentially be utilised to support large scale knowledge representation of grid.

Xing et al [77] proposed a method to describe grid environment as a whole in 2006. The key idea of the method is to design a core grid ontology and then extend it by adding additional classes, properties and constraints in order to cover comprehensive descriptions of all required grid entities. However, this method is impractical since it makes it difficult for developers to use description languages different to those used by the core ontology, to collaboratively represent individual grid entities and their relationships. There are already many schemas and ontologies developed which can be used to represent certain scopes of a grid environment. These representations are based on different description languages (e.g. Dublin Core in plain XML and RDF [78], HL7 in plain XML [54] and OpenEHR ADL [55]). If there is only one language used for representing a grid environment, this requires lots of redesign and transformation work in order to achieve the representation. If these legacies are used, in most cases it will result in massive work on developing mapping mechanisms in order to deliver the interoperability between the core ontology and descriptions of individual grid entities. It could be even worse since the changing nature of grid and knowledge representation technologies may lead to frequent and excessive changes in "this solution based" knowledge representation of a grid environment.

2.7 Summary

This chapter reviewed the traditional distributed biomedical systems, grid, healthgrid and knowledge representation technologies, which provide rational and technical foundations for this project. For grid, an overview, a grid architecture: OGSA, a grid infrastructure: WSRF, and two well-known grid middleware tools, Globus Toolkit and OGSADAI have been introduced. For healthgrid, an overview and requirements and its relationship with EHR have been presented and discussed. For knowledge representation, an overview and the Semantic grid concept have been presented. Some important work related to this project has also been discussed.

Chapter 3

Bioprofile Analysis over Grid

3.1 Introduction

This chapter aims to discuss major issues involved in delivering a grid environment for bioprofile analysis. It is organised as follows. Section 3.2 presents the concept of bioprofile, clarifies the general characteristics that a bioprofile has and discusses the requirements of bioprofile analysis. Section 3.3 overviews the framework for bioprofile analysis over grid. Section 3.4 proposes an architecture for bioprofile analysis over grid. Section 3.5 summaries this chapter.

3.2 Bioprofile and Bioprofile Analysis

3.2.1 Overview of Bioprofile

The concept of bioprofile is based on the idea of a lifelong sequence of information concerning factual events and reports relevant to a citizen's health. A bioprofile is a personal 'fingerprint' that fuses together a person's current and past medical history, biopatterns and prognosis. It combines data, analysis, and predictions of possible susceptibility to diseases.

Bioprofile is beyond the concept of today's Electronic Health Record (EHR). Figure 3.1 presents the overview of bioprofile. As can be seen, bioprofile concerns certain data categories. In general, a bioprofile may include today's EHR, multimodal data, biomarkers and other valuable information (e.g. travelling, habits and sports), where today's EHR are mainly replacements of those paper records of individual patients' medical information, such

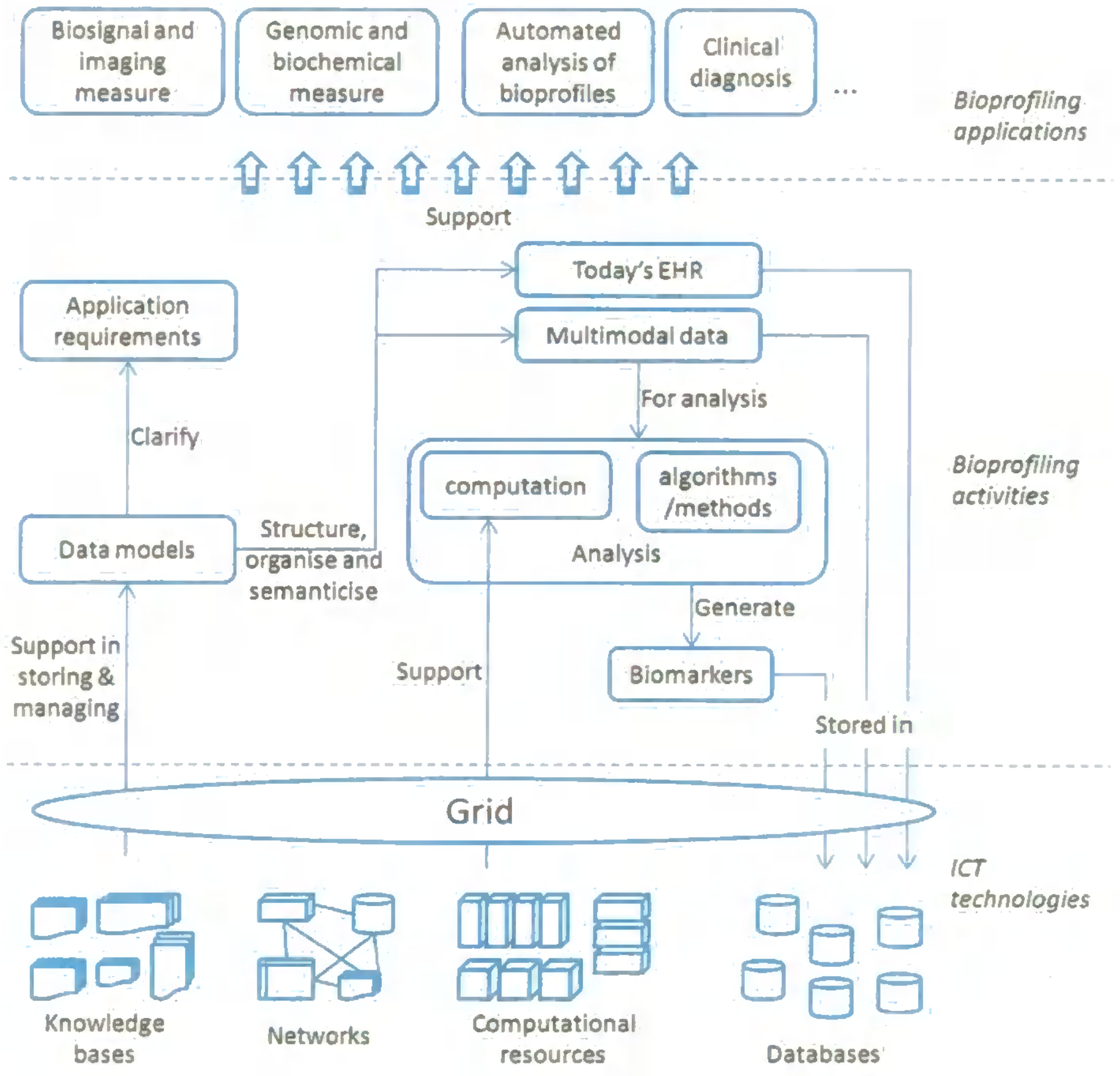


Figure 3.1: Overview of bioprofile

as, medical history, examination and progress reports of health and illnesses; multimodal data mainly include radiology images (e.g. Computed Tomography (CT)), electrophysiology (e.g. electroencephalography (EEG) and genomic data; and a biomarker can be considered as a substance used as an indicator of biomedical state. It is specific to a biomedical measurement type or a kind of analysis based on specific analysis algorithms/methods.

Bioprofile requires data models to structure, organise and, in some cases, to add semantics to EHR, multimodal data and biomarkers so that such data can be stored, retrieved, used and integrated in a standardised and manageable way. Besides these functionalities, some data models may also be developed to clarify application requirements in the aspects of clinical needs, security, ethics, quality of service, etc.

Bioprofile needs the support from ICT technologies. It requires storage facilities (e.g. file systems and relational databases) to hold EHR, multimodal data and biomarkers. It requires computation facilities (e.g. high performance cluster or just standalone PCs) to provide computation power for the analysis and the manipulation of bioprofile data. It may require knowledge bases for storing and retrieving bioprofile-related knowledge (e.g. bioprofile terminology and thesauri). It requires communication networks to provide data transmission facilities between individual IT facilities. It also requires remote resource sharing technology, such as grid, to fulfil its needs in the aspects of discovery, access, monitoring and allocation of distributed and heterogeneous IT facilities.

3.2.2 Characteristics of Bioprofile

The concept of bioprofile has many characteristics, in particular:

- A person's bioprofile may be distributed in different bioprofile centres (i.e. hospitals, surgeries and other healthcare organisations). This might be because of the mobility of a person, as the person may have lived in different regions and had health investigation in different bioprofile centres;

- Some contents of a person's bioprofile may be duplicated over two or more bioprofile centres. This can be a result of information exchange between different bioprofile centres. For example, the exchange of patient records in order to provide coherent investigation on a patient when inter-hospital transfer has to be taken place;

- A person's bioprofile contains different forms of contents. Conceptually, a bioprofile includes a person's current and past bio-history, biopatterns, diagnosis and prognosis. A biopattern further categories bioprofile contents into different types of biomedical data (e.g. genomic data, electrophysiological data and radiology images) and related biomarkers;

- Bioprofiles can be huge. A bioprofile covers all valuable data obtained from the whole life of a person. It may contain volumes of large size biomedical data (e.g. medical images). The storing of a group of bioprofiles normally requires enormous storage space;

- The same bioprofile content may be stored using different data types and formats over different bioprofile centres. For example, clinical information can be stored in either relational databases, or XML databases or text files; and Electroencephalograph (EEG) can be stored as binaries in either EDF or EEE format;

- The access to bioprofiles is restricted. This concerns issues, such as privacy, confidentiality and ethics. The restriction can have different requirements in different countries/regions.

3.2.3 Bioprofile Analysis and Its Requirements

Bioprofile analysis is a critical part of bioprofiling. It is a process that observes changes in biomarkers derived from raw health data (e.g. EEG, CT and MRI) based on analysis algorithms/methods (e.g. Fractal Dimension [79] and Tsallis Entropy [80]). It supports both long term health assessments (e.g. for early detection of the onset of diseases) and short term health assessments (e.g. early detection of onset of events, such as epileptic seizure and key changes in the depth of anaesthesia during surgery).

To carry out bioprofile analysis, there are several general requirements when considering the characteristics of bioprofile as aforementioned. First, we need access to distributed resources. The types of resources include bioprofiles, metadata for bioprofiling (i.e. the creation, use and update of bioprofiles), software implementations of analysis algorithms/methods and compute elements. In practice, metadata for bioprofiling can be further categorised into (human) subject description, biopattern metadata (e.g. description of health data and biomarkers), and metadata for analysis algorithms/methods and related software implementations. All data, knowledge and analysis components can be stored in relational databases, XML databases, binaries, text files, ontologies, etc.

There is also a requirement for federated search and integration of bioprofiling metadata. As described before, the same bioprofile content may be stored using different data types and formats. It is essential to use metadata to describe bioprofile contents. However, existing metadata used for bioprofile descriptions in different bioprofile centres are very likely to be different to each other. The difference may be the result of the use of different storage systems (e.g. XML and relational databases). It may also be due to the use of different description schemes. For example, "date" may be described in either day-month-year or year-month-day format; the definition of excessive blood pressure uses different terms, such as high blood pressure or hypertension. If a user needs to access required health data from different bioprofile centres, the metadata used in those centres must be integrated in order to display information with uniformed format to the user and/or for other systems/mechanisms for further process.

Users require different interfaces in bioprofile analysis. There are two user groups who are keen to use bioprofile analysis facilities: as clinical users and healthcare researchers. They however have different interests in healthcare. The comparison between the two groups in this aspect shows that clinical users (e.g. medical doctors) normally take more focus on the health state of a subject, but healthcare researchers are concerned more about the relationships between inputs, analysis algorithms/methods and outputs. That is to say, in most cases, clinical users prefer to carry out bioprofile analysis only based on the information of specified subject(s) (e.g. the ID of a patient). Healthcare researchers may like to give information about specified health data and/or algorithms/methods and to see what results they can get.

Security is always the most important aspect in healthcare applications. Bioprofile analysis requires access to and the processing of personal health data, which are sensitive to human health, rights and freedoms. To analyze any bioprofiles, all required security functionalities must be taken account of during the collection, the processing and the storage of any health and related data to comply with current rules and legislation.

Quality of Service (QoS) is also essential to most bioprofile analysis applications. For example, for long term bioprofile assessment, clinicians may require to carry out analysis on months or years of patient bioprofiles in near-real-time in order to get the view on the health state of a patient quickly; for short term bioprofile assessment, real-time nonlinear processing of hours or days of patient health data may be necessary to derive the biomarkers so that timely action can be taken.

3.3 Framework for Bioprofile Analysis over Grid

The framework is designed based on the consideration of the following basic bioprofile analysis over grid elements and their characteristics.

3.3.1 Basic Elements of Bioprofile Analysis over Grid

A. Bioprofiling Applications and Requirements

Bioprofile analysis requires a number of bioprofiling functionalities, such as bioprofile discovery, retrieval, integration, manipulation and analysis. The scale of a bioprofile analysis application can be either large or small. There could be different levels of bioprofile analysis applications. For example, if considering bioprofile analysis is a meta-level bioprofiling application, underneath, we may have biosignals analysis, medical image analysis, genomic data analysis, etc. Further down, we can have EEG analysis, ECG analysis, MRI analysis, CT analysis, DNA sequence analysis, etc.

A bioprofile analysis application is the reflection of a single or a group of specific needs determined by users. Such needs normally include issues in the aspects of user-defined bioprofile analysis functionality, security, Quality of Service (QoS), ethics, user interface, etc. To realise a bioprofile analysis application, specific needs will be transformed into a number of application requirements, and finally be accomplished by using appropriate mechanisms, which individually or collaboratively meet those application requirements. The form of the transformation from the user-specific needs to the requirements of a bioprofile analysis application mainly depends on application characteristics and environments (e.g. local or remote, underlying bioprofiling and ICT facilities). The requirements of a bioprofile analysis application will change when there are any changes of user needs, bioprofiling, and ICT technologies and techniques.

B. Bioprofile analysis related Resources

Bioprofile analysis related resources are those to be used for bioprofile analysis. They can be divided into two categories. One is bioprofile data. The other is bioprofiling facilities, including analysis software/tools, bioprofile data management software/tools, mechanisms of general ICT technologies (e.g. database management system, high performance cluster, security enforcements), etc. Bioprofile data are information, which can be used for analysis and integration, in order to provide evidences for e.g. clinical investigation and biomedical research. They are either inputs or outputs of bioprofile analysis. Bioprofiling facilities provide various functionalities, but are mainly used to generate, refine, retrieve, transfer and integrate bioprofile data. Bioprofiling facilities are tools for bioprofile analysis.

Most bioprofile analysis related resources are existing, but heterogeneous across different bioprofile centres. The rebuilding and/or the reorganisation of such resources into homogeneous ones in order to realise the concept of sharing is possible (e.g. build of new distributed databases with uniformed data schema to replace those old ones with different data schemas in order to share distributed data). However, in most cases, it is impractical since the rebuild and reorganisation will lead to a series of changes in bioprofiling facilities, staff retraining and disruption of healthcare services, which are time-consuming and also resource-wasting. To reach the resource sharing concept for bioprofile analysis, it is recommended to use add-on mechanisms rather than the rebuilding and/or the reorganisation of any bioprofile analysis related resources.

C. Grid

Grid is a distributed resource sharing concept, which can address those bioprofile analysis requirements resulting from distributed bioprofiles, lack of computation resources, etc. At present, the concept of grid concerns many resource sharing aspects, such as virtualisation of heterogeneous resources for sharing, resource discovery and management, data movement and related management across distributed sites, service composition and workflow management.

Many grid architectures, infrastructures and middleware tools have been proposed in the last decade, such as the grid architecture: OGSA [1], the grid infrastructures: OGSI [37] and its replacement WSRF [36], grid middleware tools: Globus Toolkit series [81], OGSADAI [41], UNICORE [44], etc. However, today's grid technology is still far from the completeness. A number of major grid issues are still under investigation, such as distributed data query, security and QoS provisioning over distributed environments, resource broker, workflow, etc. More contributions are required to realise the technology of grid.

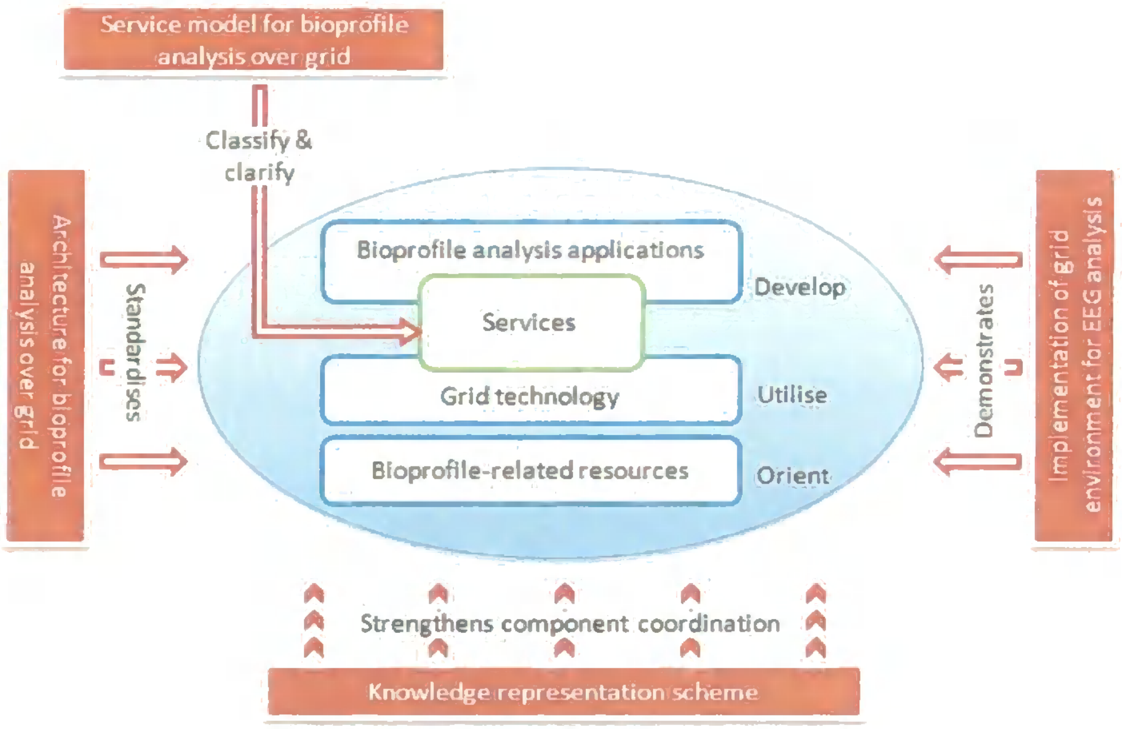

Service model for bioprofile analysis over grid
Classify & clarify
Architecture for bioprofile analysis over grid
Standardises
Bioprofile analysis applications
Services
Grid technology
Bioprofile-related resources
Develop
Utilise
Orient
Demonstrates
Implementation of grid environment for EEG analysis
Strengthens component coordination
Knowledge representation scheme

Figure 3.2: Overview of the framework for bioprofile analysis over grid

3.3.2 Framework Overview

Building a grid environment for bioprofile analysis is a long-term project. This is because of the immaturity of both bioprofile and grid technologies, and the complexity of their integration. The framework for bioprofile analysis over grid is proposed to provide standardisations of the development and use of grid environment for bioprofile analysis. The overview of the framework is presented in Figure 3.2.

As can be seen in the figure, the framework is designed based on the above-introduced three basic elements of bioprofile analysis over grid. It focuses on four areas of work,

1) an architecture for bioprofile analysis over grid, which standardises the categorisation, determination and organisation of the main components of bioprofile analysis over grid; 2) a service model for bioprofile analysis over grid , which classifies and clarifies the main categories of services in bioprofile analysis over grid ; 3) implementation of a grid environment for EEG analysis, which demonstrates how resources, grid middleware and services can be utilised, organised and implemented to support bioprofile analysis over grid; 4) a knowledge representation scheme, which semantically strengthens the global coordination between components of bioprofile analysis over grid. The four areas of work will be discussed in the remainder of this thesis.

The framework follows a general approach of building grid environment for bioprofile analysis. The approach is expected to save development time, reduce development workload and eliminate the disruption of daily healthcare services. It considers three development issues applying to those basic elements of bioprofile analysis over grid.

A. Bioprofile analysis related resources oriented

   As discussed in the previous section, the rebuilding and/or the reorganisation of bioprofile analysis related resources for the realisation of the concept of sharing is impractical since it is normally time-consuming and resource-wastful. It is thus necessary to investigate each sharing-required bioprofile analysis related resource, and to develop additional mechanism(s) (e.g. wrappers for data schema transformation) on top of it in order to virtualise and/or transform it into a (set of) "standardised" (i.e. a certain level of uniformisation) resource(s) for sharing. The virtualisation and transformation of a bioprofile analysis related resource may have different approaches, but should always be based on the characteristics of the resource.

B. Utilisation of developed grid middleware, service and any other appropriate techniques

Developing specific middleware and services, which fit into a specific distributed environment is ideal but it is normally impractical since the development requires long time and numerous human, financial and material resources. Therefore, the utilisation of developed grid middleware, service and any other appropriate techniques is important in the development of any grid environments for bioprofile analysis.

The utilisation may cover various grid-related mechanisms, such as those introduced grid middleware tools and services, and other techniques (e.g. workflow, semantics, security and QoS). The mechanisms to be utilised may be either generic or application-specific. The generic grid-related mechanisms are mainly those contributions made by grid communities, such WS-GRAM, RFT and GSI [38]. The application-specific mechanisms could be those outcomes made by any healthgrid related projects (e.g. Health-e-Child [4] and BioinfoGRID [82]).

The utilisation is a step-by-step process. Using today's grid technology it is difficult to meet most distributed bioprofile analysis requirements since many grid-related techniques are still under investigation. Moreover, some developed grid-related techniques are improved day-by-day. The utilisation thus has to follow the development of the grid technology. The introduction of any new or improved grid-related techniques must also go through several steps, as 1) investigation of developed grid-related techniques; 2) deployment of selected ones; 3) and evaluation of them.

C. Development of bioprofile analysis applications and associated services

The developed grid middleware and services can deliver a range of functionalities, which can be used to support bioprofile analysis. The support is however not comprehensive. Other functionalities/services. especially those bioprofile analysis-specific ones, still need to be investigated and developed in order to support various bioprofile analysis applications.

The development of a service can be either standalone or extentions of available grid middleware and services. A service can hold one or more bioprofile analysis-required functionalities. It may provide a (set of) new developed functionalities, or act as an interface for the access of the composited functionalities offered by those already available grid middleware and services. The development of a service must be driven by bioprofile analysis applications. The development may need to consider a number of issues, such as usability, flexibility, extensibility and reusability, in order to make the service adaptable to different bioprofile analysis applications and the continuous development of the bioprofile analysis over grid concept.

A bioprofile analysis application is the reflection of some user needs. The development of a bioprofile analysis application requires the clarification of application requirements based on user needs, making use of and/or developing bioprofile analysis-related resources, grid middleware, services and other required mechanisms to meet those clarified requirements, and also improve implementations based on user feedbacks.

3.4 An Architecture for Bioprofile Analysis over Grid

In order to clarify the main grid components for bioprofile analysis and their relationships, an architecture for bioprofile analysis over grid is proposed. The architecture contains six layers of grid components, as shown in Figure 3.3. Each layer in the architecture is a collection of related functions which delivers services/facilities to the layer above and/or other layers, and may also collect services/facilities from the layer below and/or other layers. In reality, it is not necessary to fit every system into one of the layers. But it is also not recommended to develop a closed system crossing several layers since this could cause difficulties in maintenance and extension of the system.

In the architecture, the bottom is the physical resources layer, which holds all bioprofiling facilities, including data, computation, knowledge, communication network resources, etc. These kinds of resources exists in current Information and Communication Technology (ICT) infrastructure, and have already been used in, for example holding bioprofiles, providing computation functionalities and intranet and internet connections. Example systems in this layer are 1) hardware and operating systems, which provide low level host environments for all other systems; 2) relational databases, XML databases, file systems and other storage mechanisms for storage of bioprofiles and related metadata; 3) knowledge bases for computerised collection, organisation and retrieval of bioprofile analysis related knowledge; 4) high performance clusters, high throughput clusters and middleware-enabled PCs for the support of analysis and manipulation of bioprofiles; 5) and Local Area Networks (LAN) and Wide Area Networks (WAN) for connecting all computing systems together.

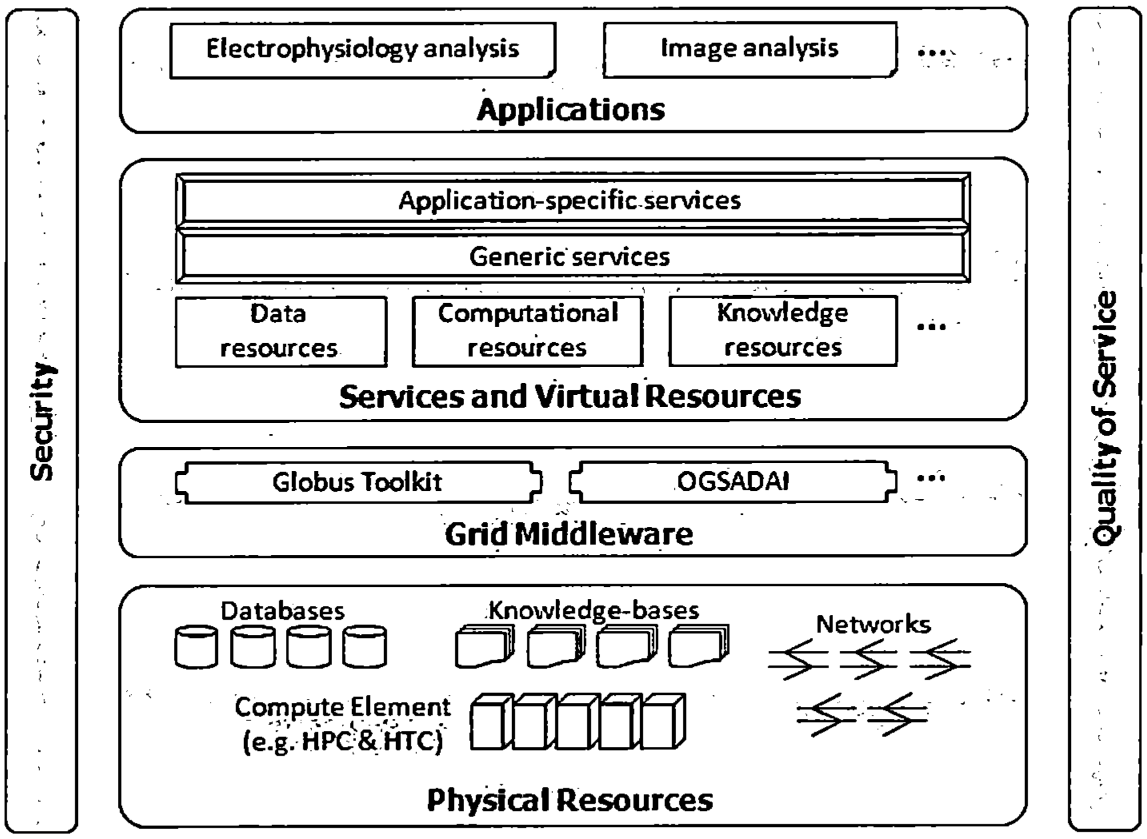


Figure 3.3: An architecture for bioprofile analysis over grid

The grid middleware layer provides functions to virtualise all real resources into virtual resources in order to provide standard interfaces for the access of distributed and heterogeneous resources. It also offers functions to, for example bridge virtualised resources together, organise virtualised resources into Virtual Organisations (VO), monitor and manage resources, and along with security and Quality of Service (QoS) mechanisms to deliver secure and application required QoS to users at grid level. Examples of grid middleware (tools) are Globus Toolkit [81], OGSADAI [41] and UNICORE [44].

The services and virtual resources layer concerns all (grid) services and virtual resources. In practice, services can be either interfaces to or representations of virtual resources. They are built for specified purposes, for example, for distributed query of bioprofiles and bioprofiling metadata or as interfaces for the access of functions provided by underlying grid middleware (e.g. job management, resource brokerage, grid information access and data transfer management). In the design, services are categorised into two levels, as application-specific and generic level services. Practically, a service can access virtualised (physical) resources either directly or indirectly (i.e. by the invocation of other services).

The applications layer contains all bioprofile analysis applications, such as Electrocardiography (ECG) analysis and MRI analysis for clinical diagnosis support. In the architecture, an application uses and constructs underlying services and virtual resources to deliver a set of functions for a group of specified requests. For example, the application: bioprofile analysis for Alzheimer's disease research may use and construct EEG analysis, Positron Emission Tomography (PET) analysis and service/resource discovery services to support its required EEG and PET analysis functions. In practice, the use and construction of services can be realised and packetised into, for example Web applications, embeddable tools and

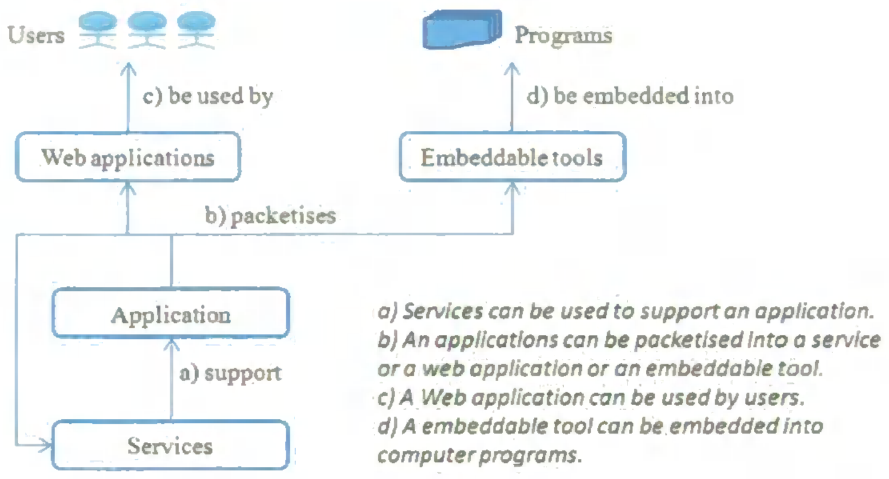

Users
Programs
c) be used by
d) be embedded into
Web applications
Embeddable tools
b) packetises
Application
a) support
Services
a) Services can be used to support an application.
b) An applications can be packetised into a service or a web application or an embeddable tool.
c) A Web application can be used by users.
d) A embeddable tool can be embedded into computer programs.

Figure 3.4: Use and construction of services for applications

services, which can then be used by end-users, programs and other higher level services and/or applications, respectively (see Figure 3.4).

Security and QoS layers contain all security and QoS policies and related enforcement mechanisms, which can be built into every layer introduced above. The rigorous security and QoS requirements in bioprofile analysis means these two layers are important in the general grid architecture. They are responsible for providing security functions (e.g. message integrity and confidentiality, authentication, and secure logging and audit) and delivering application required QoS to users, respectively.

3.5 Summary

This chapter has presented the general concept and characteristics of bioprofile, discussed bioprofile analysis requirements, overviewed the bioprofile analysis over grid framework, and proposed an architecture for bioprofile analysis over grid. The framework is designed based on the consideration of three elements, as bioprofile analysis application and requirements, bioprofile analysis related resources and grid. It focuses on four areas of work, and follows a general approach of building a grid environment for bioprofile analysis. The proposed architecture provides a standard for the categorisation, determination and organisation of core components of bioprofile analysis over grid. It contains six layers of mechanisms, which are essential in providing functionalities required by distributed bioprofile analysis applications.

Chapter 4

A Service Model for Bioprofile Analysis over Grid

4.1 Introduction

Service is a core grid component. It normally provides platform-independent protocols and standards used for exchanging data between application layer mechanisms and distributed resources. In grid, services can be either stateful or stateless. They can be constructed with various functionalities to support different applications on the access and to make use of distributed and heterogeneous resources.

This chapter aims to present a service model for bioprofile analysis over grid, which clarifies how services can be constructed and organised in order to support bioprofile analysis over distributed environments. The work here focuses more on business level services, as compared to those concerns addressed in OGSA. The chapter is organised as follows. Section 4.2 illustrates the design principle used in this work. Section 4.3 proposes a service architecture for bioprofile analysis over grid. Section 4.4 uses EEG analysis as an example application to demonstrate how to build and organise services based on existing services and grid middleware tools to support bioprofile analysis in details. Section 4.5 concludes this chapter.

4.2 Design Principle

Services are functional mechanisms to applications. To design a set of services, which can be used to support certain applications over grid, it is important to consider some issues, such as usability, flexibility, extensibility, reusability and the strategy of development. The building of services for bioprofile analysis over grid also follows several key principles, such as

- Application-driven development. The purpose of developing services is to support applications. The design of services for bioprofile analysis must be driven by those bioprofile analysis requirements, as specified in Section 3.2.3;
- Separation of concerns and layered services. This is related to the issue of flexibility, extensibility and the separation of development roles. As described before, bioprofile analysis over grid is a cross-domain concept. To realise this concept, the support of a number of functionalities is required for the access to distributed resources, data integration and the delivery of security and QoS. It is thus necessary to break the whole concept into pieces and layered structures, and to develop services with distinct features in order to make services able to be developed in parallel, to make different functions able to be updated or upgraded individually with minimum or without inter-influencing, and to make additional functions easier to be integrated;
- Generic. This is important in the issue of service reuse. In practice, a generic service is more likely to be reused than a non-generic one. Since bioprofile is an ongoing development concept, it is desirable to develop services with a generic manner so that they can be easily reused and can adapt to changing conditions and requirements of bioprofile analysis applications.

4.3 A Service Architecture for Bioprofile Analysis over Grid

4.3.1 Overview

The service architecture for bioprofile analysis over grid contains two major layers of services: generic services and application-specific services, as shown in Figure 4.1. The generic services layer contains all generic grid services, which provides grid functionalities to support application-specific layer services. In general, we divide services in this layer into nine main categorises. They are:

- Data and knowledge access services, which provide basic data query, update, transfer and knowledge discovery functionalities;
- Computation services for the access of computational resources;
- Information services for the monitor and discovery of resources and services in VOs;
- Security services, which deliver security functionalities to applications;
- QoS services, which offer mechanisms to enforce specified QoS policies in order to guarantee user required QoS;
- Distributed query services for query of data and knowledge from distributed data and knowledge resources. These kinds of services are normally built on top of data and knowledge access services;
- Job services for the generation, submission and management of grid jobs;

- Workflow services for the description and management of various application tasks to support the delivery of automated business processes in grids;
- Resource broker services for the selection and allocation of the "most appropriate" resources and services.

The application-specific services layer provides "bioprofile analysis specific" reusable functionalities to bioprofile analysis applications.The main categories of services in this layer includes,

- Bioprofile access services, which provide functionalities for bioprofile retrieval, update and integration;
- Bioprofiling metadata services, which offer bioprofiling metadata access and integration in order to support bioprofile access and analysis in an automatic manner;
- Bioprofile analysis services, which provide analysis functionalities to bioprofiles.

4.3.2 The Role of Existing Grid Middleware Tools and Services

There are some existing services and tools, which can be used to support analysis of bioprofiles over distributed environment. Most of them are low level mechanisms, which are generic to most applications. The functionalities they can offer include, but are not limited to, data access and transfer, execution management, monitor and discovery, and security.

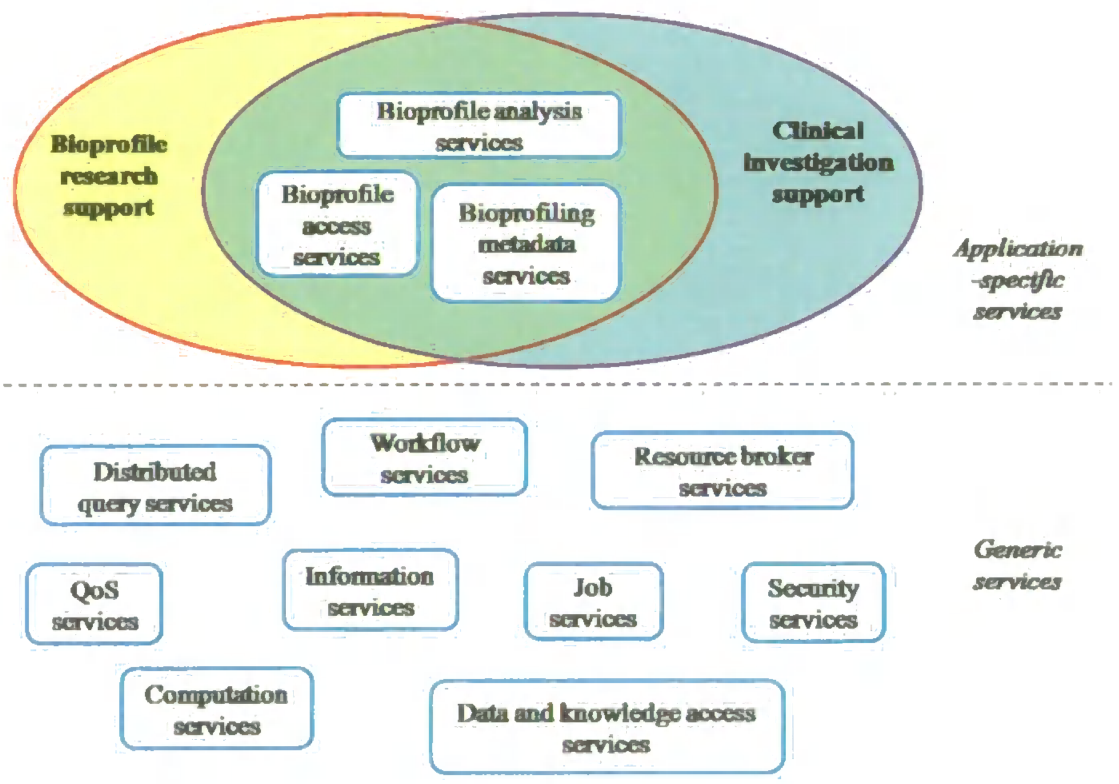

Bioprofile research support
Bioprofile analysis services
Bioprofile access services
Bioprofiling metadata services
Clinical investigation support
Application -specific services
Distributed query services
Workflow services
Resource broker services
QoS services
Information services
Job services
Security services
Generic services
Computation services
Data and knowledge access services

Figure 4.1: Categorising services for bioprofile analysis over grid

A. Data Services

Bioprofile analysis applications often need to manage, provide access to, and aggregate large quantities of data at one or many bioprofile centres and/or computation resources. Examples of services and tools, which are existing with the potential to support bioprofile movement, include GridFTP, RFT and OGSADAI, where:

- GridFTP provides libraries and tools for reliable, secure, high-performance memory-to-memory and disk-to-disk data transfers [83]. For bioprofile movement, it can be used in transmission of those bioprofiles and analysis software, which are formatted in text and binary files;
- Reliable File Transfer (RFT) service provides for the reliable management of multiple GridFTP transfers [38]. For bioprofile movement, it can handle the transmission of large numbers of bioprofile data files in a manageable and recoverable way, especially to avoid any failure of movement due to some unexpected faults occuring during transmissions;
- OGSADAI, as described in Section 2.3.3, enables data resource exposition on to grids. For bioprofile movement, it is a useful tool for the access, management and integration of heterogeneous bioprofile data resources, especially those data stored in relational and XML databases.

B. Execution Management Services

Bioprofile analysis applications normally require deploying and managing analysis executions on distributed computational resources. Grid Resource Allocation and Management (GRAM) [38] as a relevant service, addresses some of these issues, offering a Web services interface for imitating, monitoring and managing the execution of arbitrary computations on distributed computer elements. It can be integrated with other execution-related mechanisms, such as meta and local scheduler (e.g. Condor [84]) to provide a comprehensive approach to management of distributed bioprofile analysis.

C. Information Services

Most bioprofile analysis applications require mechanisms for resource/service discovery and monitoring of analysis executions/jobs in order to find relevant resources (e.g. bioprofiles, analysis software and computer elements) and to retrieve (up-to-the-second) resource properties, respectively. Monitoring and Discovery System (MDS) [85] is a tool, which realises part of these functionalities, and can be used to support the publication and retrieval of various resource and service information in a grid environment for bioprofile analysis. For instance, the use of MDS to publish database schema information on an Index services in order to let users or other services know how to access the database; and to retrieve information of registered services/resources from an Index service in order to locate services/resources which can be used for a specific analysis task, etc.

D. Security Services

Nearly all bioprofile analysis applications need security mechanisms to support the control of access to bioprofiles and software implementations of analysis algorithms/methods. There are some security tools nowadays which can be used to enable some basic security functions, such as message protection, authentication, delegation and authorisation. Grid Security Infrastructure (GSI) [38] is an example of such a protocol, which uses grid-mapfile [38] for authorisation, WS-Security [86] for message-level protection and X.509 [87] for authentication and delegation. For bioprofile analysis, it can be used to provide a solution to a certain level of access control to bioprofiles, the protection of transmission of bioprofiles between remote sites, and credential delegation for fetching data between distributed resources.

4.3.3 Application-specific Services Level Approach

A. Bioprofile Access Services

Services in this category decompose the bioprofile access problem into multiple, replaceable components, which include but are not limited to: bioprofile query; bioprofile update; bioprofile data transfer; bioprofile format transformation; and bioprofile integration, as presented in Figure 4.2.

The query, update and transfer components can be implemented with the support of underlying data access services, information services and/or distributed query services. Some data access services are interfaces or representations of bioprofile data resources. They work with grid middleware (tools) closely to provide direct or indirect access to bioprofile databases, which are built and managed by, for example, relational and XML databases, and file systems. Other data access services may offer data transmission functionality to generate and manage transfers of (large) bioprofile data over distributed bioprofile data resources.

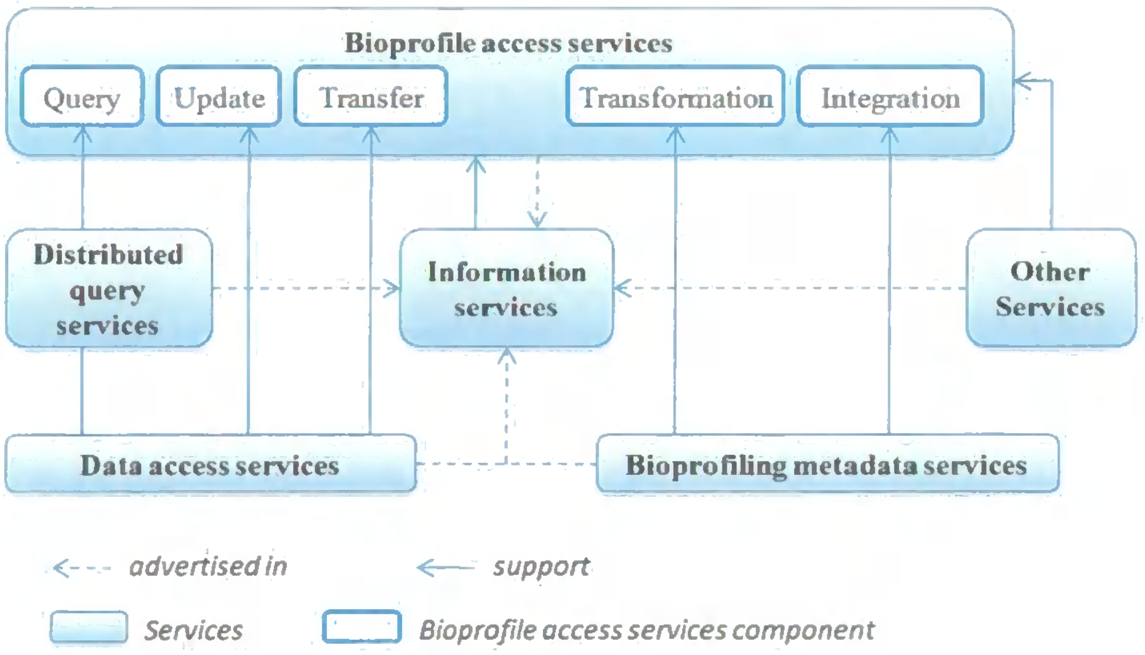


Figure 4.2: Constructing services in support of bioprofile access

The transformation and integration components are mechanisms which offer functions for transforming and integrating heterogeneous bioprofile contents into those with user/service required representation formats. They can be implemented on top of query components and with the support of underlying bioprofiling metadata services (described later). The bioprofiling metadata support may include bioprofile metadata query, retrieval of mapping schemas/ontologies, etc.

In addition, other services can also give support to the bioprofile access problem, For example, distributed replication services can deliver replication functionality in order to support the update of bioprofiles into multiple replicated data resources; Workflow services may generate and manage complex bioprofile access workflows in order to deliver dynamic use of underlying services for various bioprofile access

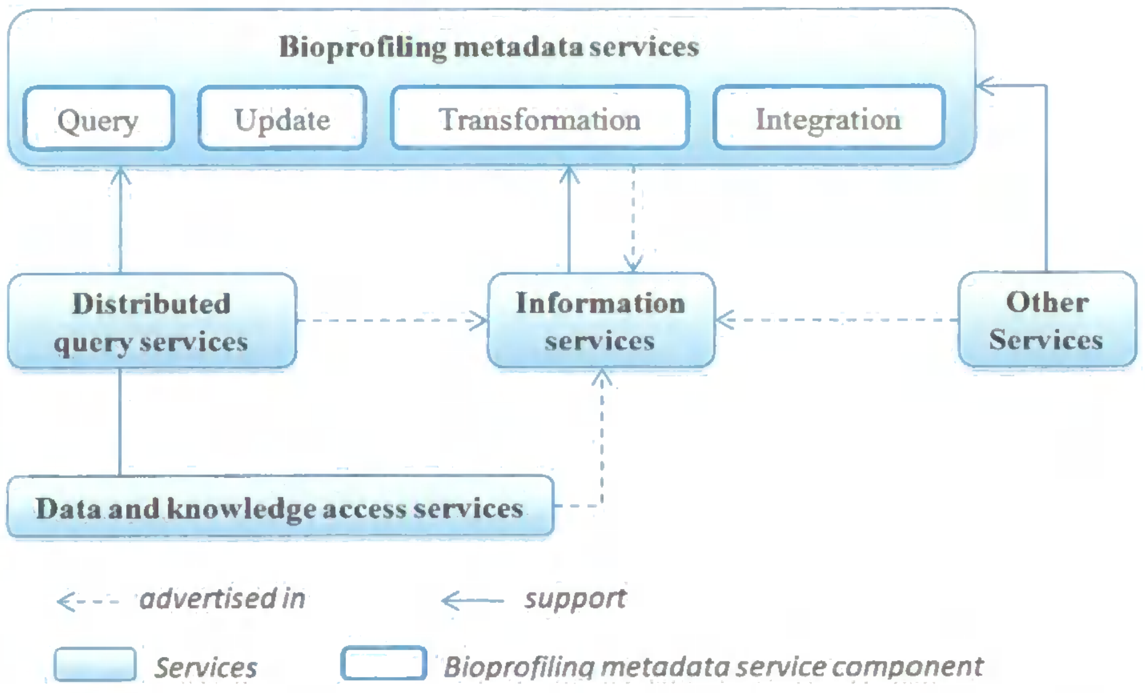

Bioprofiling metadata services
Query
Update
Transformation
Integration
Distributed query services
Information services
Other Services
Data and knowledge access services
advertised in
support
Services
Bioprofiling metadata service component

Figure 4.3: Constructing services for bioprofiling metadata access and integration

requests; Security services can realize the "restricted access" to bioprofiles; and QoS services ensure that QoS of bioprofile access can meet users' requirements.

B. Bioprofiling Metadata Services

Services in this category solve the bioprofiling metadata problem. They are implementations of certain replaceable components (e.g. bioprofiling metadata query, update, transformation and integration) for client APIs/services to access bioprofiling metadata with user/service required formats.

Metadata are data. Query components in this service category can also be implemented with the support of underlying data and knowledge access services and distributed query services, which is similar with bioprofile access services. The knowledge access here mainly includes access of the schemas/ontologies of the bioprofile data model [88] and bioprofiling metadata mappings. In practice, the knowledge access can be built on top of data access services with the support of knowledge management and inference systems (e.g. Sesame [89] and Jena [90]).

Transformation and integration components are responsible for delivering integrated and formalised bioprofiling metadata to users or bioprofile related services. In practice, they can be implemented based on built-in transformation and integration mechanisms and/or with query mechanisms, which can be used for retrieval of mapping schemas/ontologies from distributed data and/or knowledge resources.

C. Bioprofile Analysis Services

Services in this category comprise mechanisms which compose and manage lower level services (e.g. bioprofile access, bioprofiling metadata and job services) to offer generic and/or specific functionalities to support bioprofile analysis applications. In general, the main components in this service category include, but are not limited to, bioprofile access management, analysis job generation and bioprofiling metadata retrieval and matching.

The bioprofile access management component can be built on top of the bioprofile access services as described earlier. It is responsible for bioprofile access management in order to allocate the required bioprofile data (e.g. medical images and EEG) between data resources, computational resources and client ends/services based on

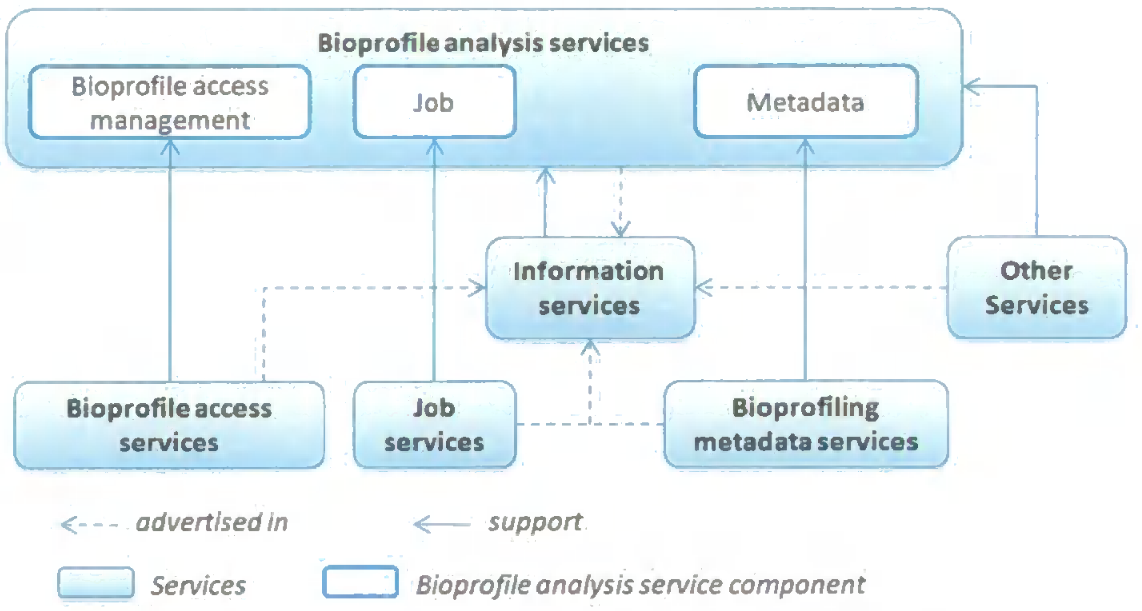

Bioprofile analysis services
Bioprofile access management
Job
Metadata
Information services
Other Services
Bioprofile access services
Job services
Bioprofiling metadata services
advertised in
support
Services
Bioprofile analysis service component

Figure 4.4: Constructing services for bioprofile analysis

specified bioprofile analysis requests.

The job component offers the high level generation and management of bioprofile analysis where implementations of analysis algorithms/methods are not embedded into bioprofile analysis services (e.g. standalone C/C++ executables and Java classes). It transforms bioprofile analysis requests into further detailed workflows and passes them to lower level job services for further processing. The management of bioprofile analysis is also abstract as it only interacts with lower level job services to monitor, schedule and destroy analysis jobs.

The metadata component contains high level mechanisms which can be set on top of bioprofiling metadata services to deliver metadata-related functionalities for bioprofile analysis applications. The main responsibility of this component is to get the description of bioprofiles, analysis algorithms/methods, etc. and to match analysis requirements with conditions in order to support the allocation of relevant resources (e.g. bioprofile data with applicable software implementations of analysis algorithms/methods) for specified bioprofile analysis applications.

4.4 Services for EEG Analysis over Grid

4.4.1 Application Requirements

Electroencephalograph (EEG) analysis is an important application in bioprofile analysis. Conceptually, Electroencephalograms are complex signals that represent the activity of the brain. EEG can serve as an objective, first line of decision support tool to improve diagnosis of most brain diseases, such as Alzheimer's disease, brain injury and Epilepsy [91].

To carry out EEG analysis, there are two special requirements in addition to those general bioprofile analysis requirements as described in Section 3.2.3. First, there is a need to match EEG records with specific implementations of EEG analysis algorithms/methods. This is because EEG records are normally stored in binary files with different formats (e.g. EEE and EDF), various numbers of channels (e.g. 21 and 25) and different lengths (e.g. 30 minutes). However, most existing EEG analysis algorithms/methods and/or their software implementations can only work with certain formats, and/or numbers of channels and/or lengths of EEG data.

Second, EEG data are sensitive to human health, rights and freedoms. EEG data, especially those EEG data files which contain patient information, may be restricted in their transfer outside of some bioprofile centres for processing. This means that EEG analysis can only be carried out within those restricted bioprofile centres. Moreover, the analysis and manipulation of EEG data may also be constrained. That is to say, analysis of some EEG data may be limited to certain agreed implementations of algorithms/methods in order to protect confidentiality.

4.4.2 A Service Model for EEG Analysis over Grid

Figure 4.5 shows a model we propose to develop services for EEG analysis based on existing grid and other ICT technologies. This model intends to demonstrate a detailed service example of bioprofile analysis, and deliver services with layered structure and certain functionalities to support EEG analysis over grid, including access of distributed bioprofiles, bioprofiling metadata integration, concurrent processing, etc. The model addresses services mainly in business level. It is planned to be modified and extended when new and stable grid techniques are introduced, such as techniques for grid security, QoS, workflow, resource broker and knowledge access.

In the model, EEG analysis metadata services, subject information services, metadata for EEG storage services and metadata mapping services are interfaces to data and knowledge resources. They can either connect to underlying databases and knowledge bases directly or be built on top of other data and/or knowledge services. They can be accessed via distributed query services in order to deliver relevant data and knowledge to bioprofile

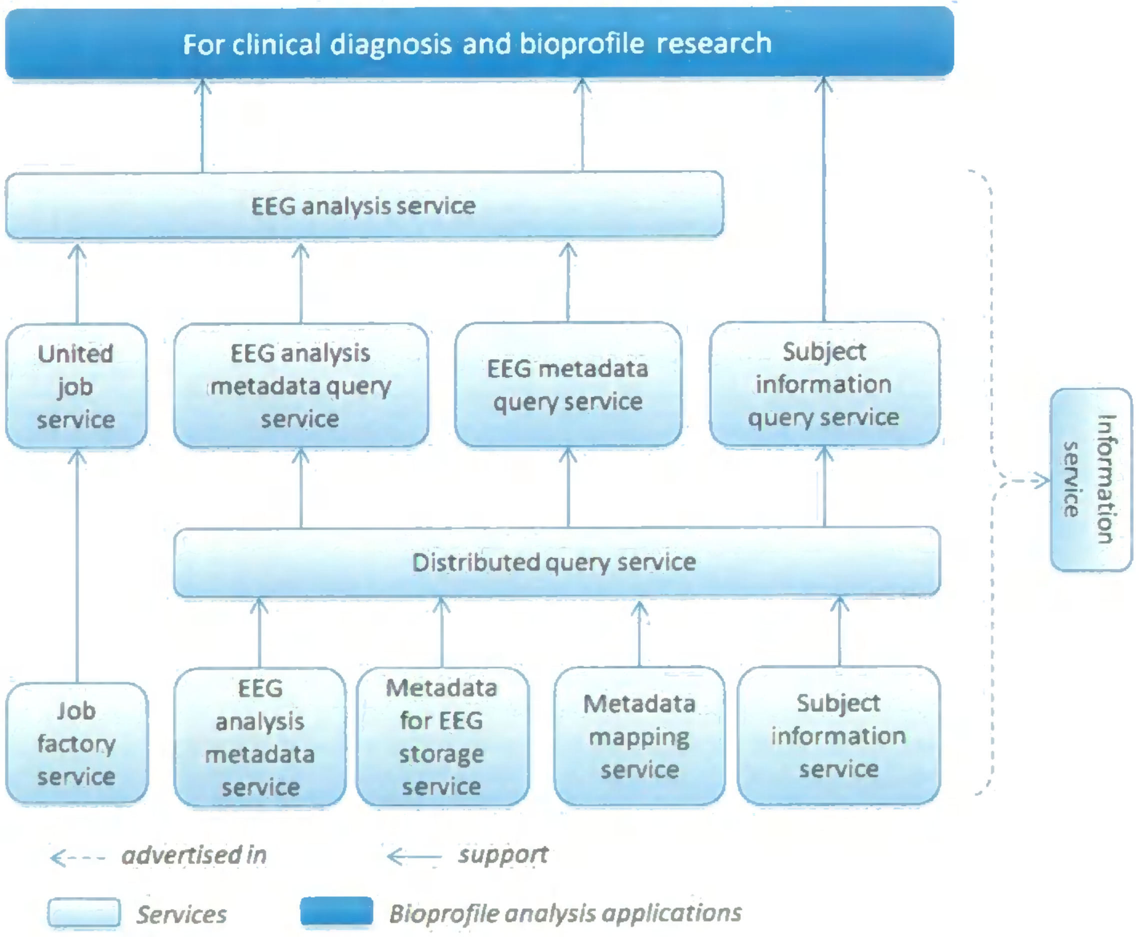

For clinical diagnosis and bioprofile research
EEG analysis service
United job service
EEG analysis metadata query service
EEG metadata query service
Subject information query service
Information service
Distributed query service
Job factory service
EEG analysis metadata service
Metadata for EEG storage service
Metadata mapping service
Subject information service
advertised in
support
Services
Bioprofile analysis applications

Figure 4.5: An EEG analysis service model

access services (e.g. subject information query services) and bioprofiling metadata services (e.g. EEG analysis metadata query services and EEG metadata query services).

EEG analysis metadata query services deal with querying of descriptions of EEG analysis algorithms/methods and related software implementations based on specified analysis tasks. They deliver these kinds of descriptions to EEG analysis services in order to let EEG analysis services know which analysis algorithms/methods are appropriate for the specified analysis requests; what relationships between specific analysis algorithms/methods and related software implementations; what and how analysis software implementations can be used for analysis to specified EEG data; what kind of results will be obtained after analysis, etc.

The distributed query service copes with distributed data querying and integration. The implementation of this category of services can be based on various approaches, such as mediated query systems, data warehouses and workflow management systems, as described in [92]. Figure 4.6 shows a workflow example of a query of EEG analysis metadata from distributed databases. In the example, a distributed query service first gets a query message from a user or a higher level service. Based on the message, it finds out information of relevant data services and resources (e.g. the data service URIs, database schemas and locations to obtain mapping schemas) from the information services. It then generates distributed query messages based on mapping schemas queried from metadata mapping services and sends those messages to distributed EEG analysis metadata services for asynchronous parallel queries. Finally, it obtains and/or passes query results to users or other services. If required, the query results will be transformed and/or integrated in order to deliver query results with a uniform format.

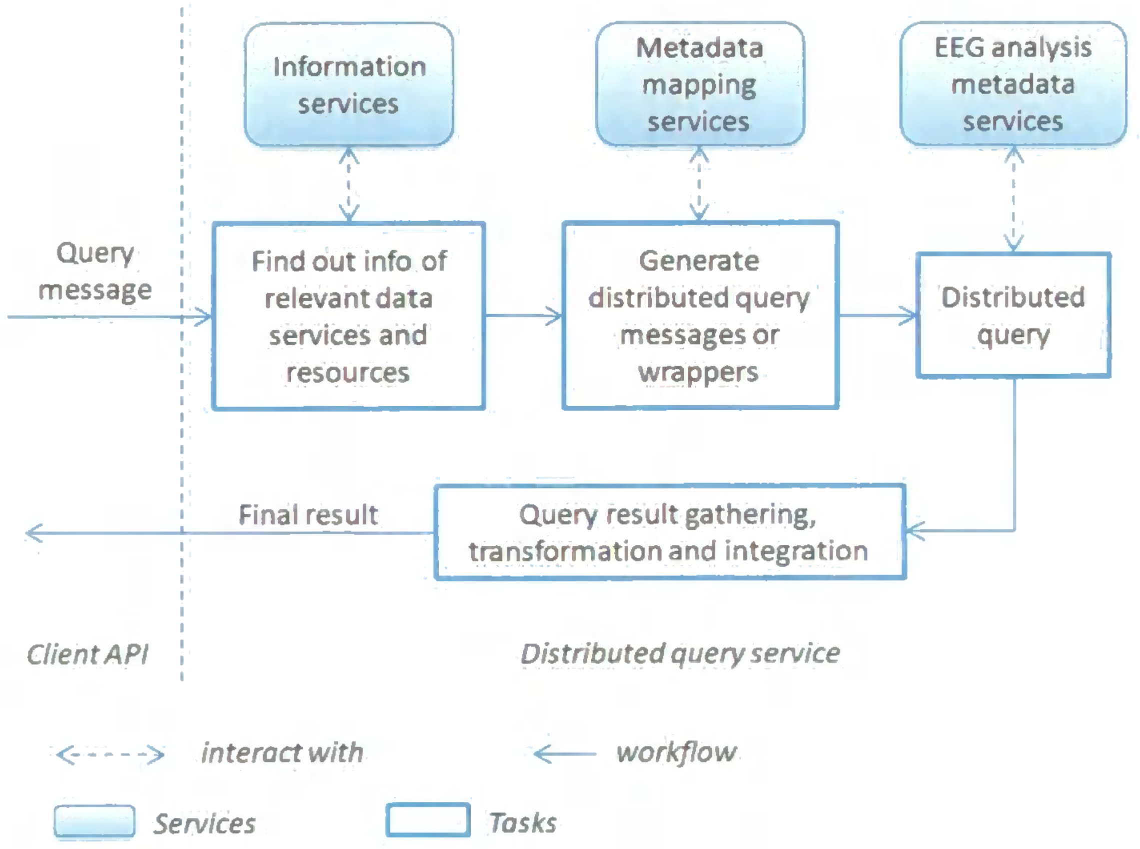

Information services
Metadata mapping services
EEG analysis metadata services
Query message
Find out info of relevant data services and resources
Generate distributed query messages or wrappers
Distributed query
Final result
Query result gathering, transformation and integration
Client API
Distributed query service
interact with
workflow
Services
Tasks

Figure 4.6: A workflow example of distributed query of EEG analysis metadata

The model separates software implementations of analysis algorithms/methods from EEG analysis services. In most cases, an EEG analysis service itself does not contain any embedded codes of analysis algorithms/methods. It only manages standalone analysis software and inputs to selected computational resources for EEG analysis. This enables flexibility in the organisation of EEG analysis, especially when EEG data access is restricted to certain locations; and/or EEG analysis is limited to certain analysis software implementations.

In the design, the EEG analysis service may use the united job service to manage multiple analysis jobs. In design, a united job service should offer simple and standard interface for users or higher level services to submit and manage analysis jobs. It should also be able to hide the complexity of using lower level job services provided by different types of grid middleware tools (e.g. gLite [45], GRAM and WS-GRAM [38]) to users.

If we take a parallel job service as a simple example of the united job service, a parallel job service should not require users/services to write down complicated job descriptions, but provide very simple information for job executions, such as the command line of running a task, the stage in file locations (i.e. locations of input files and executables/classes) and the stage out file locations (i.e. locations of output files). This kind of design may lose some flexibility of workflow construction, but utilises the independent manner of individual processes in a parallel job to maximise the usability for users and/or application developers to build up higher level services/applications. To be able to run on different types of job factory services, one solution for a parallel service is to add a layer of mechanisms to split a parallel job into multiple sub-jobs, and dispatch them to different types of job factory services for execution. Figure 4.7 illustrates a service example of using this solution for the generation and management of parallel jobs running on different sorts of grid platforms. In this example, the job splitting and sub-job generation component is the heart of the parallel job service. It provides mechanisms to get a job abstract from client API, handle the job abstract into individual tasks, encapsulate tasks into multiple sub-jobs and manage sub-jobs in order to complete the job offered by users or higher level services. The task encapsulation is based on the match between the job abstract and the obtained computational service and resource information. The sub-job description can use various job description languages for different types of grid platforms.

Like all other kinds of bioprofile analysis service, the EEG analysis service also has two major user groups, clinical users and healthcare researchers. This means that an EEG analysis service will be more usable if it has a client API, which can serve both of the groups. To achieve this, we believe it will be better to provide several levels of choice of analysis specification in the client API. For example, a user can specify a software implementation of an algorithm for analysis if he/she wants to compare several versions of implementation for performance and/or quality evaluation; or a user (e.g. healthcare researcher) may only tell an EEG analysis service the name of an algorithm he/she wants to use and lets the service decide which version of algorithm software can be used for analysis (e.g. latest and random). Alternatively a user (e.g. clinician) knows nothing about analysis algorithm, but wants EEG analysis service to help him/her to find out all available algorithms in order to carry out analysis and give a comprehensive view on a set of specific EEG files. In this model, we consider three use cases, as analysis based on a

- Known algorithm name;
- Known algorithm ID;
- Specific software implementation of EEG analysis algorithms/methods.

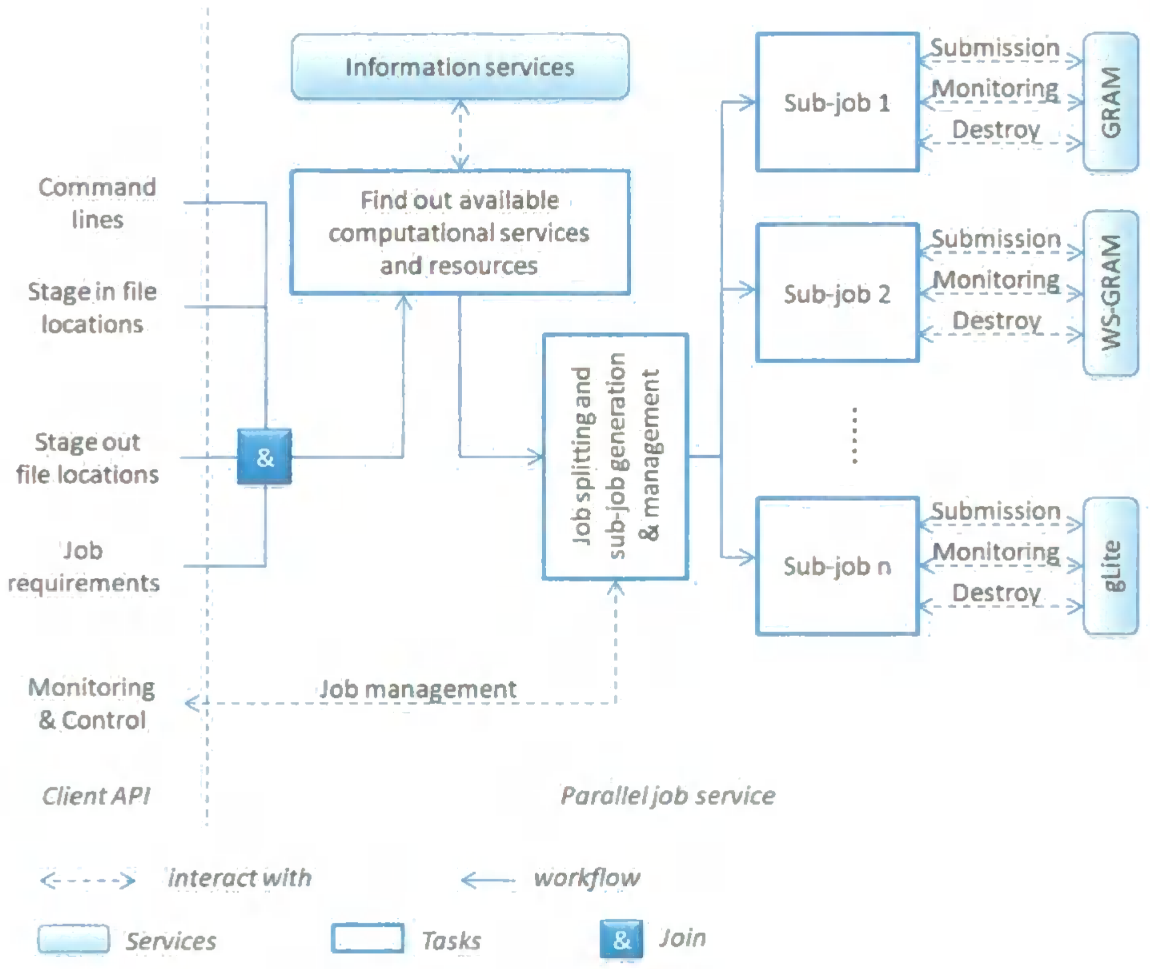


Figure 4.7: An example of using parallel job service for parallel job generation and management

Figure 4.8 presents a basic workflow example, which can be implemented within an EEG analysis service. In this example, the input parameters that must be given in order to provide essential information for analysis include 1) one of the three use cases, 2) the specified EEG file information (i.e. IDs to locate EEG files or descriptions of EEG files own by the user) 3) and the preferred locations for delivery of analysis results. The EEG analysis metadata query service is responsible for supporting the querying of EEG analysis related information (e.g. finding out analysis algorithm ID by given algorithm name, and the discovery of the relevant analysis software by given algorithm ID and EEG file format). The EEG metadata query service provides functions for the querying of EEG file descriptions by given the EEG file ID. The parallel job service generates and manages parallel analysis job based on the description (including locations) of the software implementation of the analysis algorithm, EEG file and result storage.

To make the example more understandable, we assume an EEG analysis service obtains three input parameters from client API for an analysis request: the name of an analysis algorithm, the IDs of several EEG files and a location for the result storage. The corresponding workflow for this analysis request might be described as,

a. Invoke the EEG analysis metadata query service to get the ID of the latest stable version of the name-given algorithm by default, at the same time asynchronously invoke the EEG metadata query service to get the locations and the formats of those EEG files based on the given file IDs;

b. Use the algorithm ID to find out the relevant software implementations of the algorithm, and use the obtained EEG file format information to select the most suitable implementations;

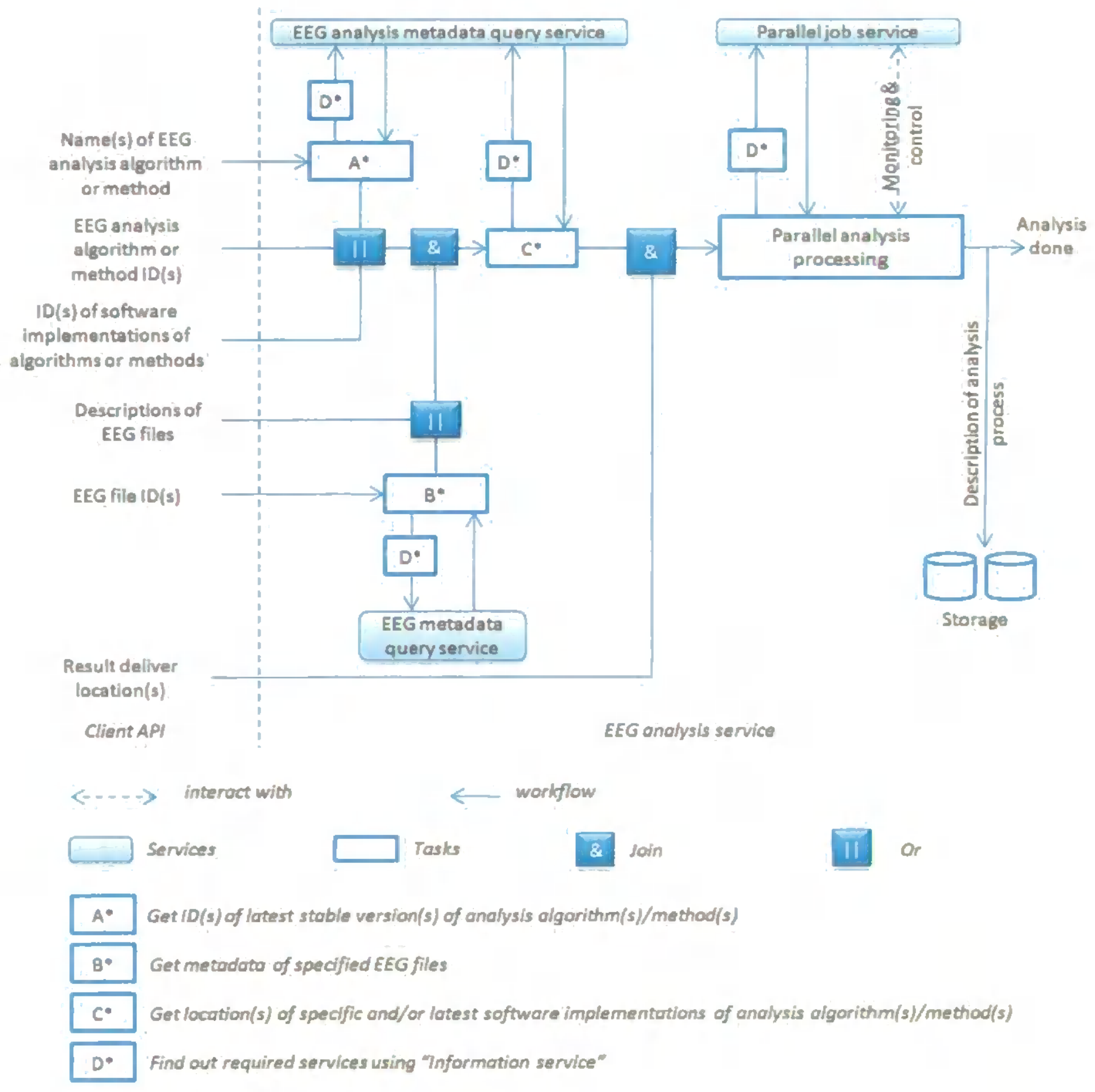

EEG analysis metadata query service
Parallel job service
D*
A*
D*
D*
Monitoring & control
Name(s) of EEG analysis algorithm or method
EEG analysis algorithm or method ID(s)
&
C*
&
Parallel analysis processing
Analysis done
ID(s) of software implementations of algorithms or methods
Description of analysis process
Descriptions of EEG files
EEG file ID(s)
B*
D*
Storage
EEG metadata query service
Result deliver location(s)
Client API
EEG analysis service
interact with
workflow
Services
Tasks
Join
Or
A* Get ID(s) of latest stable version(s) of analysis algorithm(s)/method(s)
B* Get metadata of specified EEG files
C* Get location(s) of specific and/or latest software implementations of analysis algorithm(s)/method(s)
D* Find out required services using "Information service"

Figure 4.8: A workflow example of EEG analysis

c. Generate an analysis job abstract, and send it to the parallel job service to realise the analysis process on distributed computational resources;

d. Monitor and control analysis process by contacting parallel job service. When the analysis process completes, results are delivered to the specific storage location. The whole analysis process might be described and stored in specified locations.

4.5 Summary

In this Chapter, a service model for bioprofile analysis has been proposed. The service model addresses a service design principle, a service architecture for a bioprofile analysis over grid, and an EEG analysis service model. The service design principle discusses important issues in developing and organising services in support of bioprofile analysis applications, such as usability, flexibility, extensibility, reusability and the strategy of development.

The service architecture classifies and clarifies services required in grid-enabled bioprofile analysis. The architecture has two layers: a generic services layer and an application-specific services layer. It details how to utilise contributions made by the grid community to provide a well-formed infrastructure in support of those generic functionalities (e.g. data access, job management and security). The architecture also addresses the general approach in building and managing bioprofile analysis specific services in the realisation of the concept of bioprofile analysis over grid.

To demonstrate the concept of the service architecture, an example application: EEG analysis has been employed. An EEG analysis service model has then been proposed to show how to utilise existing services and grid middleware tools, and how to develop and manage services based on those existing contributions in order to realise EEG analysis over a distributed and heterogeneous environment.

Chapter 5

Implementation of A Grid-enabled EEG Analysis Platform

5.1 Introduction

To demonstrate the concept of bioprofile analysis over grid, a grid-enabled platform has been developed to support an important bioprofile analysis application: EEG analysis, over a distributed and heterogeneous resource sharing environment.

This chapter presents how resources, grid middleware and services can be utilised, organised and implemented to support distributed EEG analysis for early detection of dementia. The chapter is organised as follows. Section 5.2 describes a scenario, which addresses the need of using grid technology to support clinical investigation on a patient due to his mobility and shortage of local computational power as well as analysis software. Section 5.3 presents two grid test-beds, which have been developed to provide essential experimental platforms for research in bioprofiling over grid. Section 5.4 presents the implementation of a grid environment, which realises the analysis scenario. The details of the implementation architecture and infrastructure will be given. Section 5.5 presents obtained implementation results and evaluates those results in the aspects of utility, collaborativity, extensibility and performance. Section 5.6 concludes this chapter.

5.2 Scenario: EEG Analysis over Grid for Early Detection of Dementia

Dementia is a neurodegenerative cognitive disorder that affects mainly elderly people [93]. At present, several acetyl cholinesterase inhibitors could be administered for dementia of the

Alzheimer's type, but for maximum benefits early diagnosis is important. Currently several objective methods are available that may support early diagnosis of dementia. Amongst others, the EEG which measures electrical activities of the brain offers the potential for an acceptable and affordable method in the routine screening of dementia in the early stages. Using current clinical criteria, delay between the actual onset and clinical diagnosis of dementia is typically 3 to 5 years. A limitation of current objective methods is that diagnosis is largely based on group comparisons, i.e. attempting to separate individuals into groups (Normal, AD, Parkinson's, etc.). An alternative to this is individualized care through subject-specific biodata analysis. Such an approach would allow us, for example, to compute biomarkers which over time would represent the subjects 'bioprofile' for dementia, and to look for trends in the 'bioprofile' that arise over time to detect possible on-set of dementia [79].

Figure 5.1 illustrates the life of a fictitious individual called Mike who was born and studied in Finland, lived in Greece and has been working in the U.K for two decades. At the age of 60, Mike is diagnosed with probable AD. To provide accurate diagnosis, his GP in the UK requires his past and present medical information (bioprofiles) which could be located in databases in several different countries (e.g. UK and Greece). Additionally, information stored in the databases would be very large as these are Mike's lifetime medical records such as EEG, clinical information, etc. Furthermore, analysis of the data would usually entail the use of complex algorithms which could take several hours to complete and could be held at various bioprofile centres. Using grid, to provide seamless access to geographically distributed data and high computational resources for complex analysis and data storage, more accurate and efficient diagnosis may be achieved.

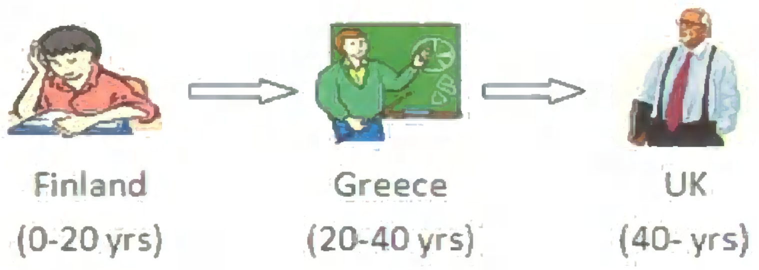


Figure 5.1: Mike's life journey

To illustrate the concept of EEG analysis over grid for early detection of dementia, a hypothetical patient pool consisting of 400 subjects, each with three EEG recordings was created. These data are hypothetical representation of recordings taken at three time instances akin to longitudinal studies carried out in reality. Each dataset consists of 21 channels of recording and is 1.3Mbytes. The recording duration is 4 minutes and the sampling rate is 128 Hz. The Fractal Dimension [79], Hjorth [94] and Tsallis Entropy [80] algorithms are used for EEG analysis.

5.3 Grid Test-beds

5.3.1 BIOPATTERN Grid

The BIOPATTERN grid is a grid platform. It is expected to facilitate secure, reliable, efficient and seamless sharing of geographically distributed bioprofile databases and support for analysis of biopatterns and bioprofiles to combat major diseases such as brain diseases and cancer. The development of the BIOPATTERN grid focuses on how to utilise and build upon existing grid, bioprofile and other relevant technologies to realise the concept of bioprofiling over grid. The BIOPATTERN grid is a collaborative research activity under the BIOPATTERN project. Its development follows the 20-year bioprofile development plan proposed by the BIOPATTERN project community [9].

The prototype of the BIOPATTERN Grid aims to provide a platform for clinical users and healthcare researchers within the BIOPATTERN Consortium to share bioprofile data, computation and analysis resources to facilitate the analysis, diagnosis and care for brain diseases and cancer.

Currently, the prototype connects five sites -the University of Plymouth (UoP), UK; the Telecommunication System Institute (TSI), Technical University of Crete, Greece; the University of Pisa (UNIPI), Italy; Synapsis S.r.l. (Synapsis), Italy, and Tampere University of Technology (TUT), Finland (see Figure 5.2). Each site holds grid nodes and may also hold bioprofile databases, high throughput cluster (e.g. Condor pool), high performance cluster, algorithms pool, Web portal, or an interface to remote data acquisition networks.

In the prototype, bioprofile databases at present contain basic patient's clinical information (e.g. weight and the status of Alzheimer's disease), EEG data (awake EEG at resting state) for dementia, and EEG data (MVEP) for brain injuries. The data are distributed over bioprofile databases at TUT, TSI, and/or UOP. The algorithms pool, which is located at the UOP site, includes software implementations of bioprofile analysis algorithms for brain diseases, such as the Fractal Dimension (FD) and Independent Component Analysis (ICA) algorithms. In addition, UoP contains GT4-based grid nodes, a condor pool (PlymGRID [95]) with up to 1,400 staff and student PCs and a Web server (i.e. the BIOPATTERN Grid Portal [96]), which holds all implemented Web applications for end-users to seamlessly access underlying grid services and resources in support of bioprofiling. Basic security mechanisms, the Grid Security Infrastructure (GSI) [97], has been enabled in the prototype.

Between UOP, TSI and TUT, there are two demonstration applications, which have been developed on the BIOPATTERN Grid for clinical diagnosis support of dementia and brain injury. UNIPI and Synapsis are connected to the BIOPATTERN Grid via a grid node based on GT4. At UNIPI, the crawling services [98] are being adapted to the BIOPATTERN Grid. Synapsis is developing an interface between the BIOPATTERN grid and the (remote) wireless acquisition network for automated remote data acquisition [99].

5.3.2 PlymGRID

PlymGRID [95] is a campus grid environment, which currently provides high throughput computing facilities to support various research and teaching activities for all students and staff at the University of Plymouth. To date, it has utilised up to 1,400 student and staff PCs to provide computation power, and the university local network to distribute computational jobs to any of available PCs.

Figure 5.3 presents the overview of PlymGRID. As can be seen in the figure, PlymGRID uses Condor [84] as the main high throughput computing middleware solution to provide job queuing mechanisms, scheduling policy, priority scheme, resource monitoring, and resource management for all computational jobs running on it. PlymGRID, at present, contains a job manager for centralised job management and resource allocation and certain job submitters for job generation and submission. Those job submitters are mainly used by advanced users, where those users normally require running those jobs which have complex workflows. Some

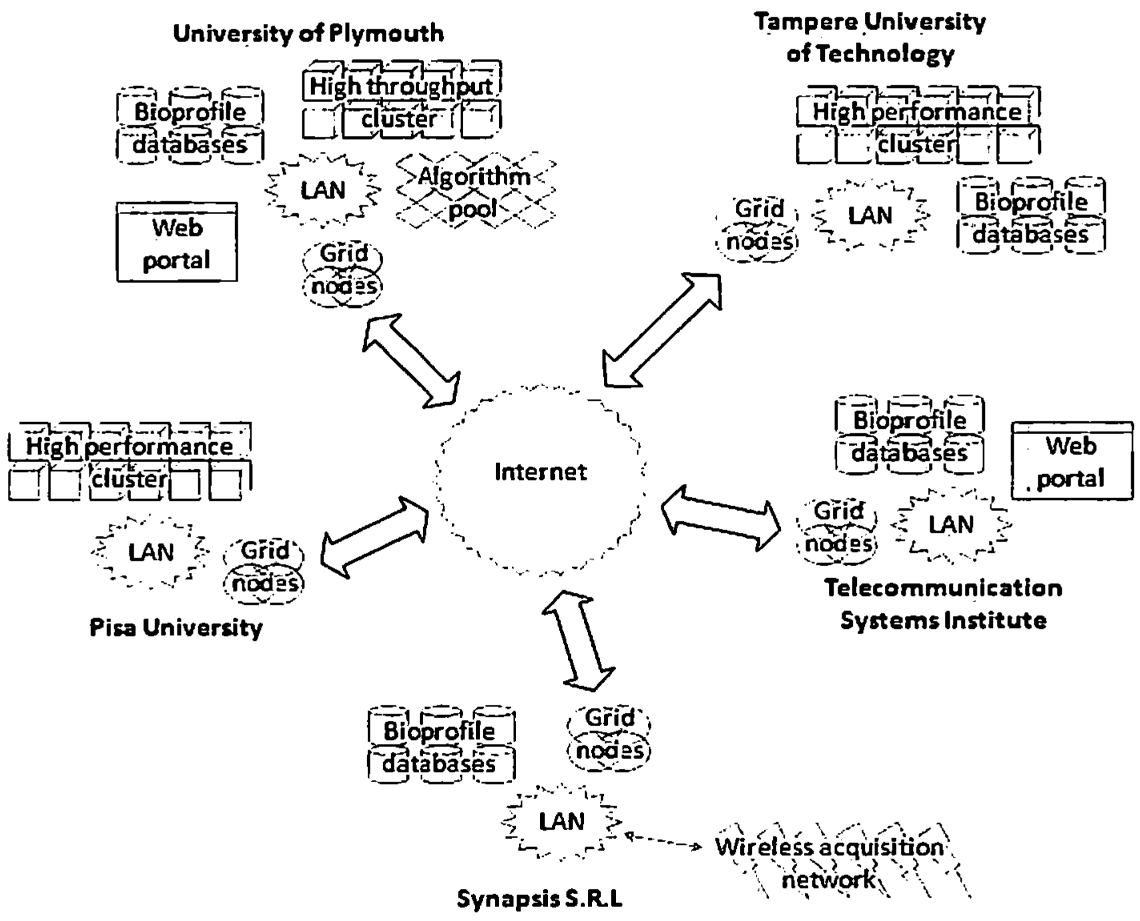


Figure 5.2: The prototype of the BIOPATTERN grid

of them are also used to bridge PlymGRID with the BIOPATTERN grid. For any other users, PlymGRID is accessible via a developed Web portal, which connects a dedicated job submitter, and allows permitted users to submit parallel jobs via Web browsers to PlymGRID and retrieve results from it. The supported program binaries currently include C/C++ executables, Java classes and compiled Matlab (C) executables.

PlymGRID is an important contribution to bioprofiling over grid. Its development is not only to provide a high throughput computing resource, but also to deliver a campus grid network with other institutes (i.e. connecting with other campus grids, such as OXGrid, Reading Grid and Bristol Grid), which can be used to support various bioprofiling research activities between different universities. As the build-up is generic, it thus benefits research in other fields as well, such as the comparison between coding algorithms for space communications [100].

5.4 Implementation Architecture and Infrastructure

5.4.1 Overview

To realise the EEG analysis scenario (see Section 5.2), a distributed EEG analysis platform has been developed based on the BIOPATTERN grid and PlymGRID. This platform aims to deliver functionalities to clinical users for seamless access and use of distributed and heterogeneous EEG analysis related resources (e.g. EEG data, analysis software and

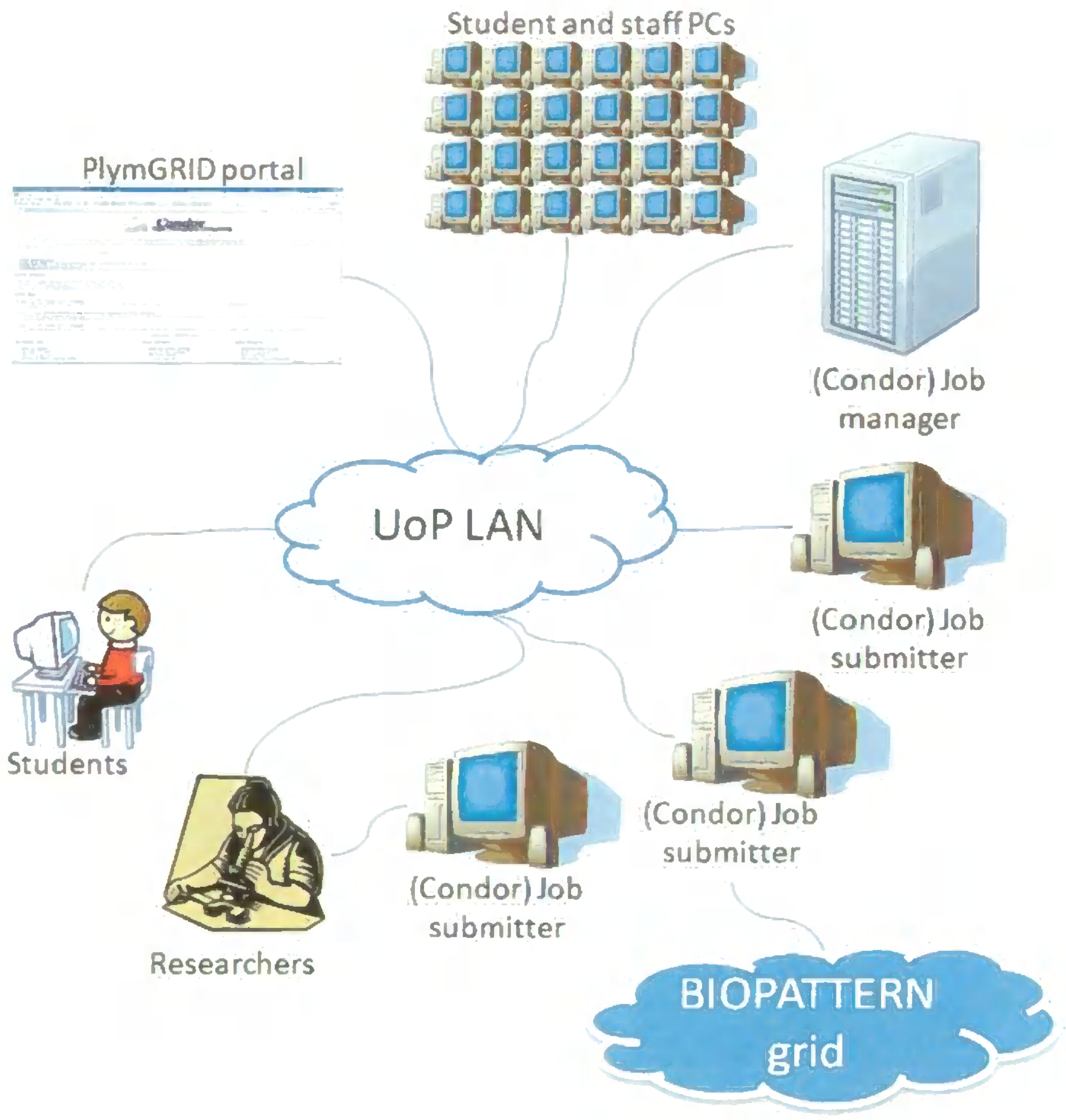


Figure 5.3: Overview of PlymGRID

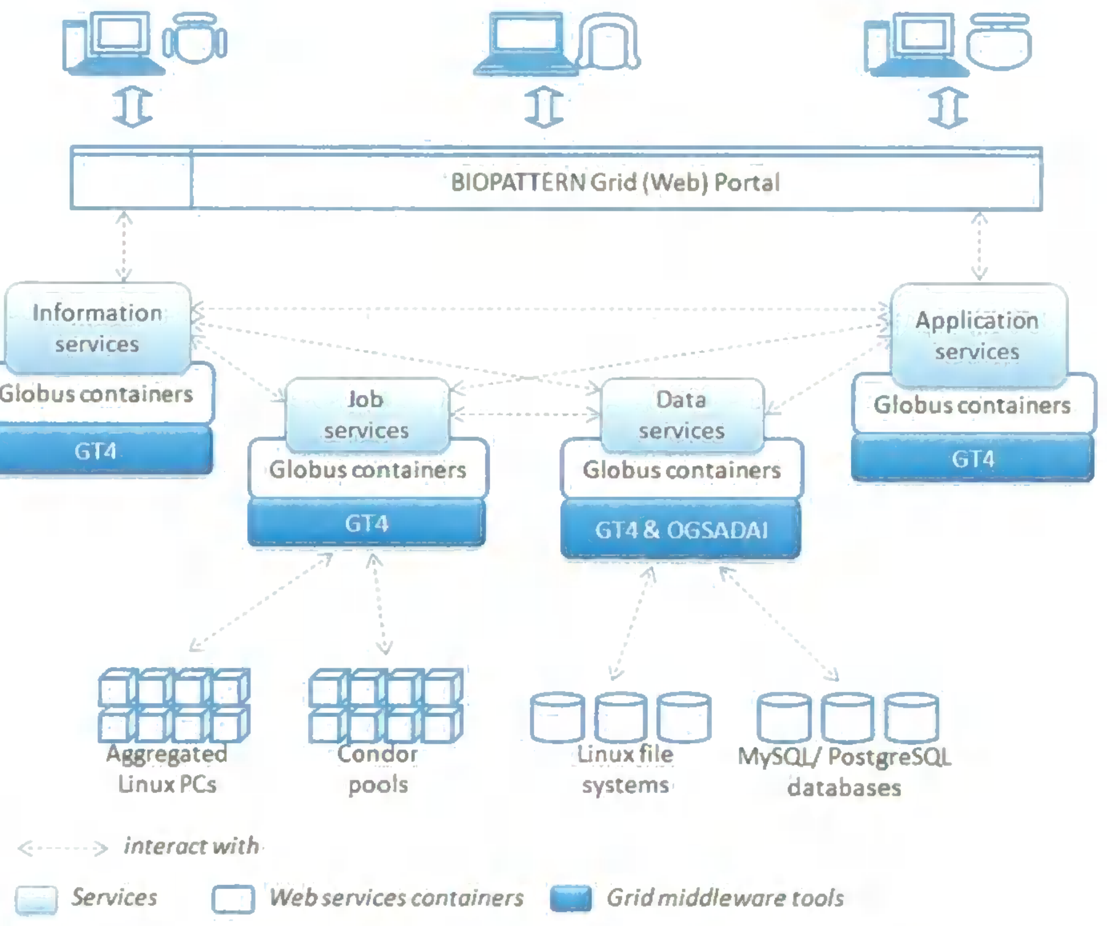


Figure 5.4: Overview of implementation of EEG analysis over Grid

computation clusters) with the support of basic security enforcements. The platform consists of implementations of grid mechanisms in the Physical Resources layer, the Grid Middleware Layer, the Services and Virtual Resources Layer, the Applications Layer and the Security Layer. Figure 5.4 presents the overview of its implementation.

5.4.2 Physical Resources Layer

This layer composes of several categories of resources, including EEG data, bioprofiling metadata, software implementations of analysis algorithms, computation elements, networks, host environments, etc.

A. EEG Data

The EEG signals were stored in binary files with EEE format. Each EEG dataset contains 21 channels, 4 minutes of recording. The size of an EEG dataset is around 1.3 Mbytes. The 400 EEG datasets were distributed to UoP (200 datasets), TUT (120 datasets) and TSI (120 datasets), respectively. Each site has 20 duplicated datasets. All EEG datasets were stored in file systems.

B. Software Implementation of Analysis Algorithms

The Fractal Dimension (FD) [79], Hjorth [94] and Tsallis Entropy [80] algorithms were employed in experiments for EEG analysis. The FD algorithm was first implemented using C and then compiled into a C executable with file size at 155 Kbytes. The Hjorth algorithm was first implemented using Java and then compiled into a jar file with file size at 7.7 Kbytes. The Tsallis Entropy algorithm was first implemented using Matlab and then compiled into a C executable with file size at 10 Kbytes and a Matlab library constructer with file size at 43 Kbytes. Corresponding Matlab Runtime Library was deployed into all computation elements.

C. Bioprofiling Metadata

The bioprofiling metadata include metadata for EEG data files, analysis algorithms and related software implementations, and mapping schemas. Except mapping schemas, all bioprofiling metadata were stored in relational databases. Examples of schemas used in experiments are presented in Appendix A. Mapping schemas were implemented using XML, and stored in file systems. In implementation, only different attribute names were used to express different metadata schemas for the description of the same type of object at the three sites for simplicity (e.g. the use of EEG duration and EEG length for the expression of recorded EEG duration, respectively). An example of used mapping schemas can be found in Appendix B. In addition, those mapping schemas were registered in mapping schema registry(s), which is/are XML documents linking head information of each mapping schema with corresponding mapping schema (file) location. An example of used mapping schema registries can be found in Appendix C.

D. Computation Elements

There were two sets of aggregated Linux PCs (four P4-3GHz-CPU 1GB-RAM machines at UoP and eight P4-3.2GHz-CPU 1GB-RAM machines at TUT) and a Condor pool (up to fifty P4-2.8GHz-CPU 512M-RAM Windows machines at UoP) used as computation elements in experiments.

E. Host environments

SuSE 9.x Linux operating systems (kernel 2.6.x) [101] were used in storing text and binary files and holding all software. MySQL (v4.1 and v5.0) [10] and PostgreSQL (v7.4) [11] databases were used in holding all bioprofiling metadata except mapping schemas.

5.4.3 Grid Middleware Layer

This layer contains two main grid toolkits, GT4 (v4.x) [38] and OGSADAI-WSRF (v2.x) [41]. For GT4, GridFTP and Reliable File Transfer (RFT) service were used for file transmission and related management, (Web Service) Grid Resource Allocation Management (WS-GRAM) for execution management, WS Monitoring and Discovery System (WS-MDS) for resource and service monitor discovery, and Grid Security Infrastructure (GSI) mechanisms to provide basic security functionalities (e.g. authentication and authorisation). OGSADAI, together with GT4 core, was used for exposition of data resources (e.g. relational databases for storing of bioprofiling metadata) and the support of federated data discovery.

5.4.4 Services and Virtual Resources Layer

The services and virtual resources layer realises the EEG analysis service model for clients to access those physical resources seamlessly. The service implementation focuses on how to build additional services (e.g. the EEG analysis service and the parallel job service) on top of existing services (e.g. the GT4 Managed Job Factory Service) to support EEG analysis over grid. All services are implemented in the (Grid) Web service style. We use Grid Development Tools (GDT) [102] for service implementation, and a Globus container to hold all implemented as well as existing services.

A. GT4 Services

Three categories of GT4 services were used to support the implementation, where Reliable File Transfer service, together with GridFTP, were used for managed transmission of text and binary files; WS-GRAM related services (e.g. Managed Job Factory Service and Managed Job Service) were used for execution management; and WS-MDS related services (e.g. Default Index Service) were used for service registry and collection of resource property information.

B. Relational Database Access Services

These kinds of services have been implemented to expose relational databases as a sort of data resources on to grids, and also to provide interfaces for access to those resources. They were built directly upon relational databases which hold bioprofile information and bioprofiling metadata with the support of OGSADAI-WSRF and GT4 Core. They include a:

- Subject Information Service for access to subject (e.g. patient) personal and clinical information;
- EEG File Metadata Service for access to metadata which describe those dummy EEG datasets;
- EEG Analysis Algorithm Metadata Service for access to metadata which describe the three EEG analysis algorithms;
- EEG Analysis Software Metadata Service for access to metadata which describe the software implementations of the three EEG analysis algorithms.

All these services were registered in GT4 Default Index Services. Related resource properties published in those index services include data resource ID and database schema.

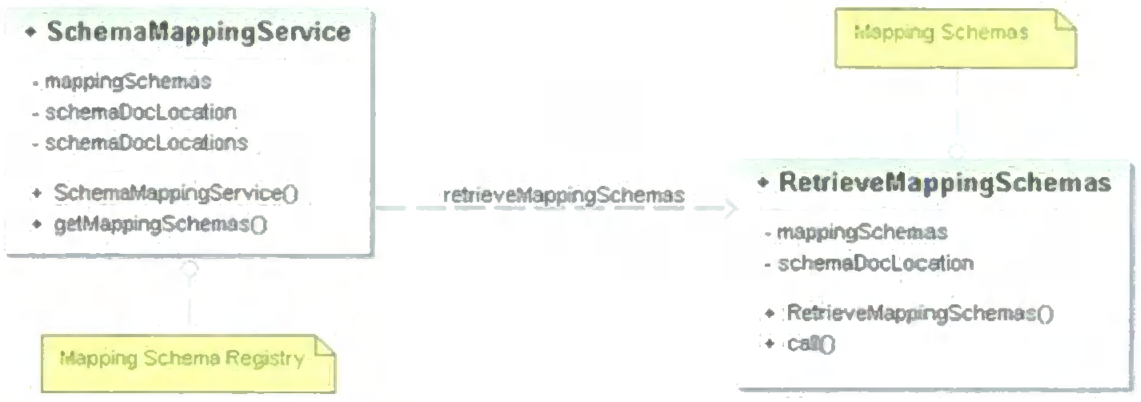


Figure 5.5: Implementing mapping schemas access service

C. Mapping Schemas Access Service

This service was developed to provide a kind of mechanism for access to mapping schemas in order to support, in this case, the integration of distributed bioprofiling metadata, which use different metadata schemas. They are linked to defined mapping schemas (see Section 5.4.2), and were built based on GT4 core. Figure 5.5 shows a class diagram, which presents the way of implementing this service, where SchemaMappingService class gets the required schema document name from the client API, then uses the RetrieveMappingSchemas class to load the schemas and finally return the loaded schema as a Document Object Model (DOM) object back to clients.

The GT4 Default Index Service was used for the registry of this service. Related resource properties published in those index services are mainly mapping schema registry documents in order to let client APIs find and retrieve required mapping schemas.

D. Distributed Query Service

This service was developed to provide distributed query functionality in a generic way. It was built upon relational database access services and a mapping schema access service. Figure 5.6 shows its class diagram, which summaries how it works. The corresponding service workflow is:

(a) The DistributedQueryService class gets a query message from a client, where the query message is SQL-like, e.g. "SELECT x FROM y WHERE z";

(b) The DistributedQueryService class decomposes the query message into "selections", "virtual table/catalog name", and "conditions";

(c) The DistributedQueryService class employs the FindDatabaseResource class to find out related data resources and services from one or more GT4 Default Index Services in parallel based on the "virtual table/catalog name";

(d) The DistributedQueryService class employs the FindMappingSchema class to find out related mapping schema resources and services from one or more GT4 Default Index Services in parallel based on both the "virtual table/catalog name" and the discovered data resource IDs;

(e) The DistributedQueryService class employs the QueryMessageGeneration class to map the query message into certain distributed query messages. Those distributed query messages follow the data schema defined by each data resource. The LoadMappingSchema class is used to retrieve mapping schemas from those mapping schema resources in parallel;

(f) The DistributedQueryService class employs the Query class to query distributed data resources in parallel based on those distributed query messages. The query results are then returned to DistributedQueryService;

(g) The DistributedQueryService class employs the ResultTFAndIG class to transform those query results into an uniform format, then deletes those replicated

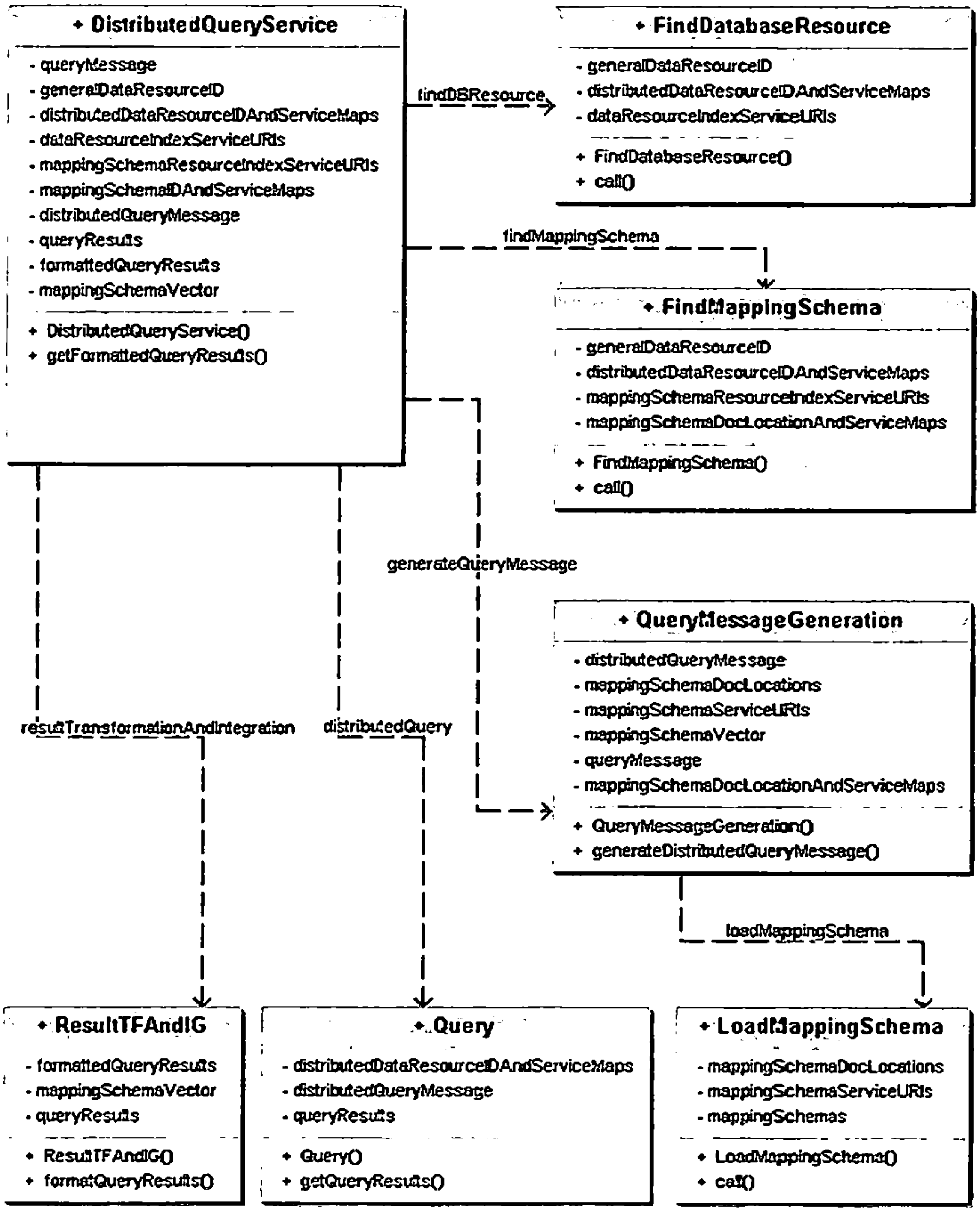

+ DistributedQueryService
- queryMessage
- generalDataResourceID
- distributedDataResourceIDAndServiceMaps
- dataResourceIndexServiceURIs
- mappingSchemaResourceIndexServiceURIs
- mappingSchemaIDAndServiceMaps
- distributedQueryMessage
- queryResults
- formattedQueryResults
- mappingSchemaVector
+ DistributedQueryService()
+ getFormattedQueryResults()
findDBResource
+ FindDatabaseResource
- generalDataResourceID
- distributedDataResourceIDAndServiceMaps
- dataResourceIndexServiceURIs
+ FindDatabaseResource()
+ call()
findMappingSchema
+ FindMappingSchema
- generalDataResourceID
- distributedDataResourceIDAndServiceMaps
- mappingSchemaResourceIndexServiceURIs
- mappingSchemaDocLocationAndServiceMaps
+ FindMappingSchema()
+ call()
generateQueryMessage
+ QueryMessageGeneration
- distributedQueryMessage
- mappingSchemaDocLocations
- mappingSchemaServiceURIs
- mappingSchemaVector
- queryMessage
- mappingSchemaDocLocationAndServiceMaps
+ QueryMessageGeneration()
+ generateDistributedQueryMessage()
resultTransformationAndIntegration
distributedQuery
loadMappingSchema
+ ResultTFAndIG
- formattedQueryResults
- mappingSchemaVector
- queryResults
+ ResultTFAndIG()
+ formatQueryResults()
+ Query
- distributedDataResourceIDAndServiceMaps
- distributedQueryMessage
- queryResults
+ Query()
+ getQueryResults()
+ LoadMappingSchema
- mappingSchemaDocLocations
- mappingSchemaServiceURIs
- mappingSchemas
+ LoadMappingSchema()
+ call()

Figure 5.6: Implementing distributed query service

query results, and finally combines those query results together into an XML document;

(h) The DistributedQueryService returns the XML document back to the client.

In addition, the GT4 Default Index Service was also used for the registry of this service.

E. Application Specific Query Services

These services were built up to provide application specific query interfaces and mechanisms to transform query requests into query messages, which are compatible with underlying distributed query services. The implemented services include the EEG metadata query service, the EEG analysis metadata query service, and the subject information query service. GT4 Default Index Service was used for registry of these services.

F. Parallel Job Service

This service was developed to provide parallel job management functionality in a generic way. It was built upon GT4 ManagedJobFactoryService and can be extended to other job/execution services. Figure 5.7 shows its class diagram, which summaries how it works. The corresponding service workflow is:

(a) The GridJobManagementService class gets three pieces of information from a client, as "command lines" that the client uses to run each task, "stage in file URIs" (i.e. locations of input files and executables/classes), and "stage out file URIs" (i.e. locations where output files will be delivered to);

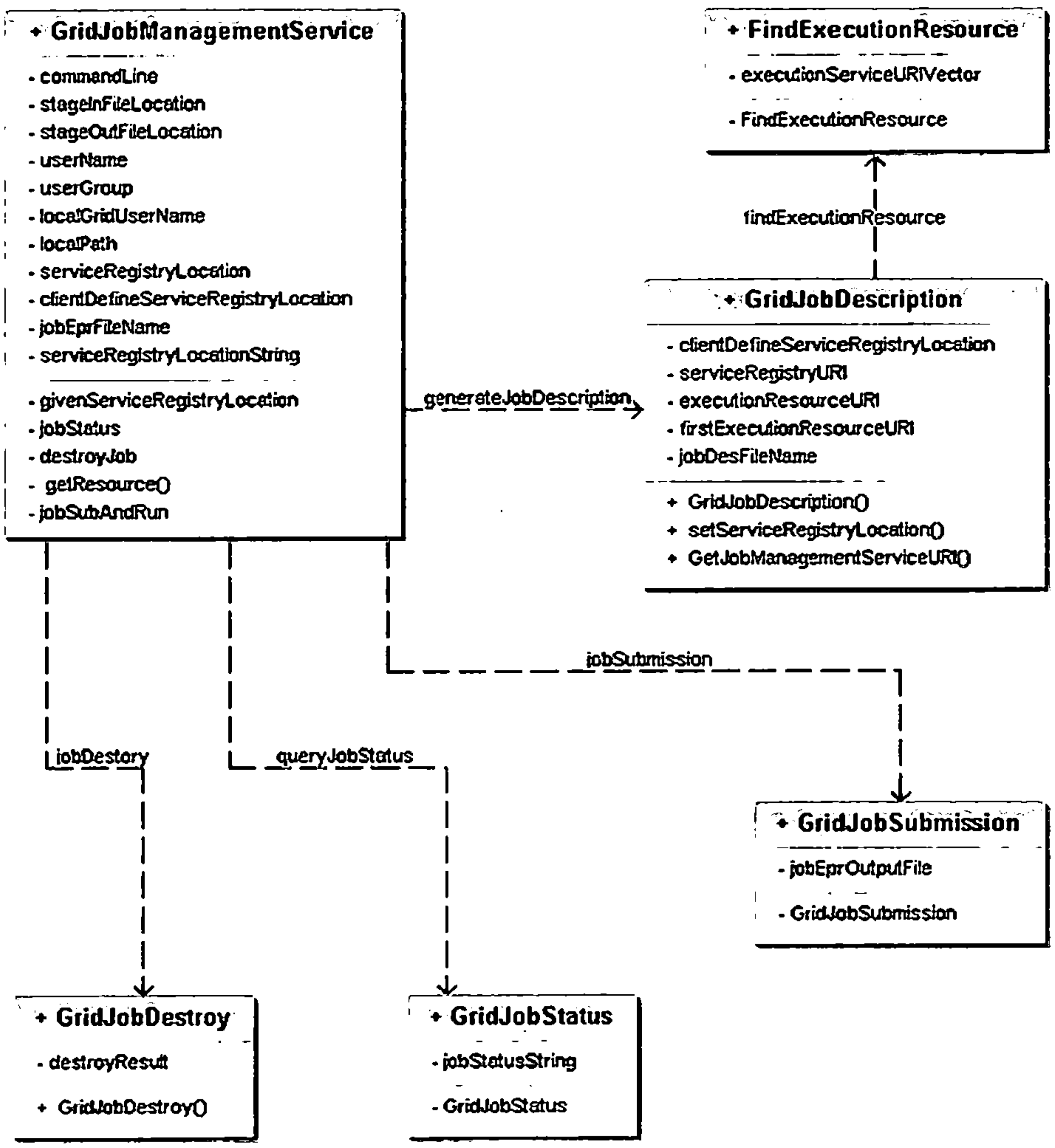


Figure 5.7: Implementing Parallel Job Service

(b) The GridJobManagementService class employs the GridJobGeneration class to generate a job, which contains all the tasks the client wants to run. The generation is based on those located execution resources (in this case ManagedJobFactoryService) and the three pieces of information obtained from the client;

(c) The GridJobManagementService class employs the GridJobSubmission class to submit the job to the distributed execution resources, obtains a job End-Point Reference (EPR) as the reference to this job, and returns the job epr to the client;

(d) The client can use the job epr to get latest job status using the GridJobStatus class via the GridJobManagementService class;

(e) The client can use the job epr to destroy the job using the GridJobDestroy class via the GridJobManagementService class.

In addition, the GT4 Default Index Service was also used for registry of this service.

G. EEG Analysis Service

This service was developed to support EEG analysis. It realises the EEG analysis service example, as addressed in Section 4.4.2. Figure 5.8 presents its class diagram, which outlines the major components of this service, where:

- The QueryEEGAnalysisAlgorithmID class deals with when a client wants to carry out EEG analysis, but he/she only knows the algorithm name(s). The EEG analysis metadata query service was used to support its query on algorithm ID(s);

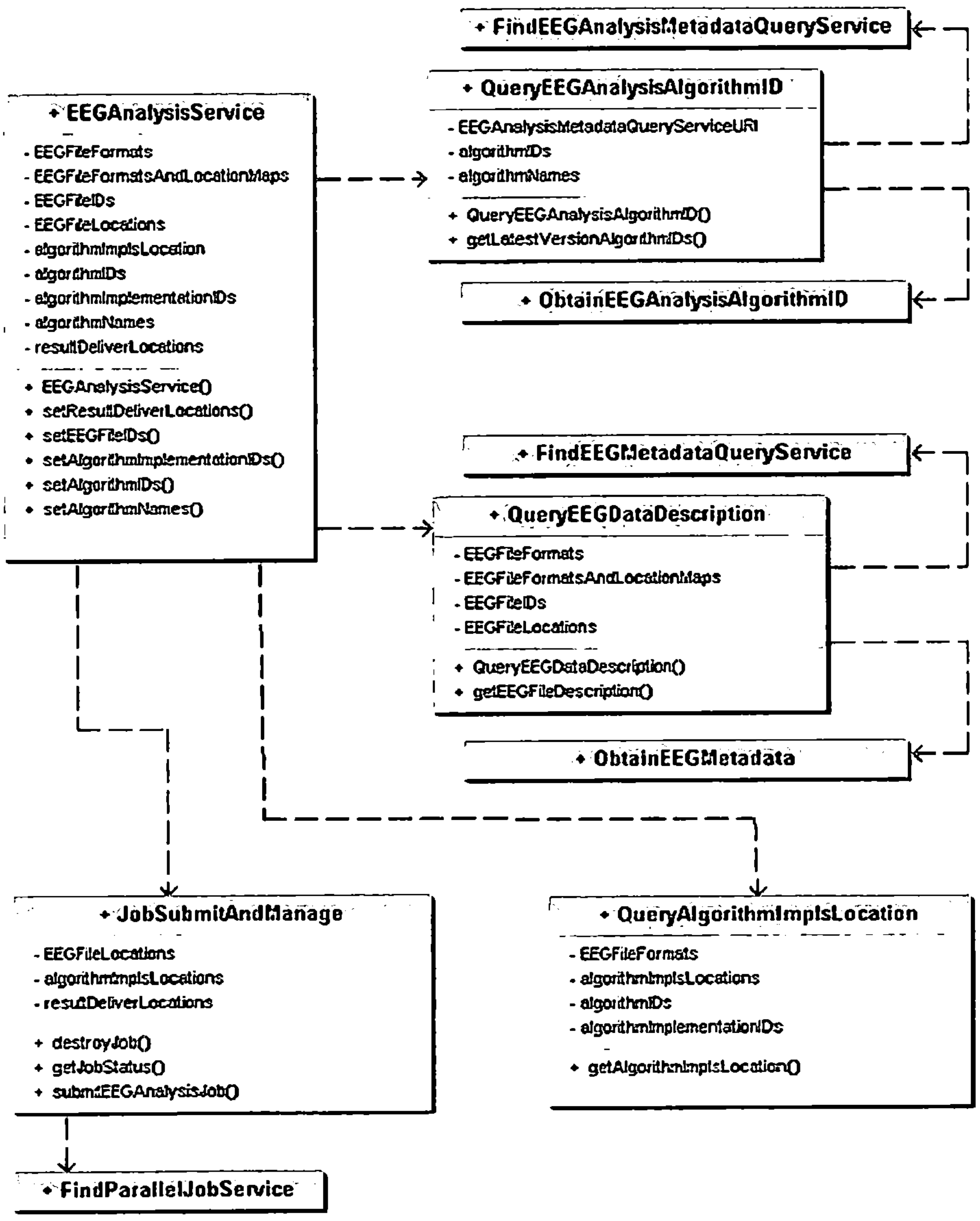
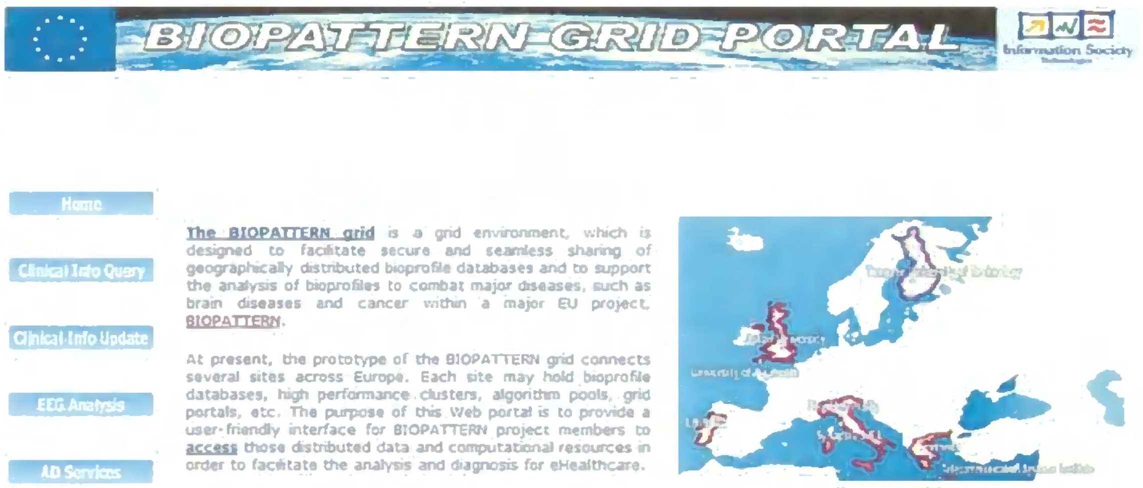

+ FindEEGAnalysisMetadataQueryService
+ QueryEEGAnalysisAlgorithmID
- EEGAnalysisMetadataQueryServiceURI
- algorithmIDs
- algorithmNames
+ QueryEEGAnalysisAlgorithmID()
+ getLatestVersionAlgorithmIDs()
+ ObtainEEGAnalysisAlgorithmID
+ EEGAnalysisService
- EEGFileFormats
- EEGFileFormatsAndLocationMaps
- EEGFileIDs
- EEGFileLocations
- algorithmImplsLocation
- algorithmIDs
- algorithmImplementationIDs
- algorithmNames
- resultDeliverLocations
+ EEGAnalysisService()
+ setResultDeliverLocations()
+ setEEGFileIDs()
+ setAlgorithmImplementationIDs()
+ setAlgorithmIDs()
+ setAlgorithmNames()
+ FindEEGMetadataQueryService
+ QueryEEGDataDescription
- EEGFileFormats
- EEGFileFormatsAndLocationMaps
- EEGFileIDs
- EEGFileLocations
+ QueryEEGDataDescription()
+ getEEGFileDescription()
+ ObtainEEGMetadata
+ JobSubmitAndManage
- EEGFileLocations
- algorithmImplsLocations
- resultDeliverLocations
+ destroyJob()
+ getJobStatus()
+ submitEEGAnalysisJob()
+ QueryAlgorithmImplsLocation
- EEGFileFormats
- algorithmImplsLocations
- algorithmIDs
- algorithmImplementationIDs
+ getAlgorithmImplsLocation()
+ FindParallelJobService

Figure 5.8: Implementing EEG analysis service

- The QueryEEGDataDescription class is responsible for getting the metadata of EEG files in order to provide location information to data access services/mechanisms for file retrieval, and to provide EEG descriptive information to this service for finding appropriate software implementations of analysis algorithms, which can work with the format used by each specific EEG file. EEG metadata query service was used to support its query on EEG metadata;
- The QueryAlgorithmImplsLocation class is used to get location information of software implementations of the EEG analysis algorithms based on either algorithm IDs or algorithm implementation IDs. It was also supported by the EEG analysis metadata service;
- The JobSubmitAndManage class interacts with the parallel job service for job submission and management based on the obtained information of EEG files, analysis algorithms and related software implementations.

5.4.5 Applications Layer

To support the EEG-based early detection of dementia application, several Web applications have been developed, which compose specific services for end-users to use underlying grid facilities via user friendly GUIs. For instance, the application employs the GT4 Default Index Service for the discovery of relevant application services; the Subject Information Query Service (application service) to get information of a specific subject; and the EEG analysis service (application service) for the analysis of all EEG data the subject has. In the implementation, JSP (v2.0) [103] and HTML (v4.01) [104] are the main programming language used at this layer. Tomcat (v5.0.17) [105] is the Servlet container employed for



The BIOPATTERN grid is a grid environment, which is designed to facilitate secure and seamless sharing of geographically distributed bioprofile databases and to support the analysis of bioprofiles to combat major diseases, such as brain diseases and cancer within a major EU project, BIOPATTERN.

At present, the prototype of the BIOPATTERN grid connects several sites across Europe. Each site may hold bioprofile databases, high performance clusters, algorithm pools, grid portals, etc. The purpose of this Web portal is to provide a user-friendly interface for BIOPATTERN project members to access those distributed data and computational resources in order to facilitate the analysis and diagnosis for eHealthcare.

Figure 5.9: The BIOPATTERN Grid (Web) Portal

holding all the implemented Web applications. Figure 5.9 shows the home page of the Web portal.

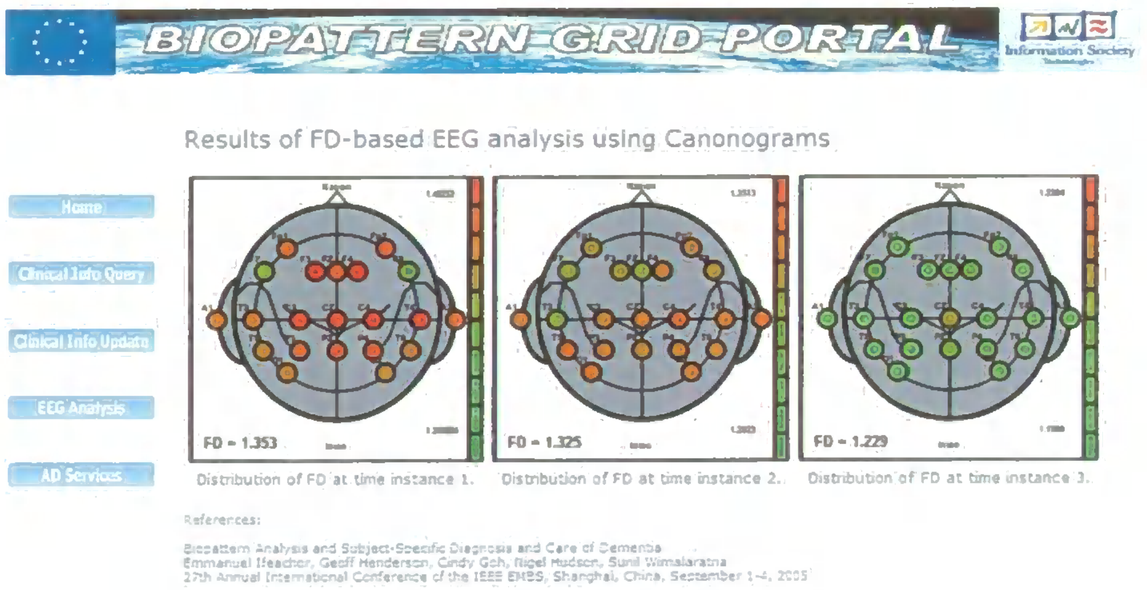
5.4.6 Security Layer

The implementation in this layer follows the Grid Security Infrastructure. Four basic functionalities have been realised, including transport level message protection, X.509 and related GT4 mechanisms based authentication and delegation; and gridmap-file based authorisation.

5.5 Results and Evaluation

5.5.1 Utility

The implementation realises the EEG analysis scenario, as described in Section 5.2. Through the Web portal, an end-user (e.g. Mike's GP) can find Mike's information and perform an analysis on Mike's EEG records using the analysis algorithms. Upon submission, Mike's information, including his previous medical records which are at TSI and TUT is retrieved and analysed. Results are then returned to the user under the implementation settings. Obtained results can be visualised using, for example, the canonograms (see Figure 5.10) and 2D expression of Tsallis entropy index (see Figure 5.11) to show changes in the EEGs indicating Mike's conditions. In Figure 5.10, the canonograms (from left to right) show the FD value (or index) of Mike's EEG taken at time instances of 1 (data at TUT), 2 (data at TSI) and 3 (data at UoP) respectively. The FD value for the left canonogram indicates Mike in a normal condition with high brain activity, whereas the FD value for the right canonogram indicates Mike in a probable Alzheimer Disease with low brain activity. The middle one shows the stage in between. Similarly, in Figure 5.11, Tsallis entropy values extracted from Mike's EEGs also show the disease progression and indicate abnormality at the time instance 3. In summary, the changes (or trends) in the FD and Tsallis entropy values provide some indication on the disease progression. This can help clinicians to detect dementia at an early stage, to monitor its progression and its response to treatment.



Results of FD-based EEG analysis using Canonograms

Distribution of FD at time instance 1.

Distribution of FD at time instance 2.

Distribution of FD at time instance 3.

References:

Biopattern Analysis and Subject-Specific Diagnosis and Care of Dementia
Emmanuel Ifeachor, Geoff Henderson, Cindy Goh, Nigel Hudson, Sunil Wimalaratna
27th Annual International Conference of the IEEE EMBS, Shanghai, China, September 1-4, 2005

Figure 5.10: Canonograms showing the distribution of FD values from EEG analysis for patient 'Mike'

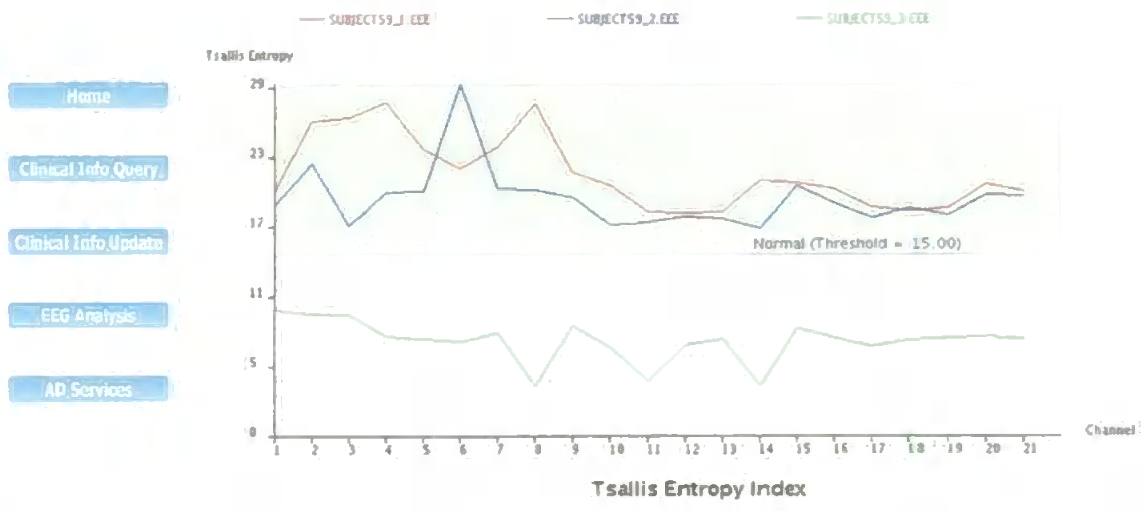


Figure 5.11: The distribution of Tsallis entropy values from EEG analysis for patient 'Mike'

5.5.2 Collaborativity and Extensibility

A layered and componentised services development strategy enables the collaborative development of grid environment for bioprofiling. The implementations of services and Web applications are distributed over UoP, TSI and TUT. For instance, lower layer service developers who are familiar with GT4 and OGSADAI but not bioprofile applications virtualise data and computational resources at UoP, TSI and TUT; higher layer service developers at UoP compose lower layer services together into for example, the EEG analysis service to support bioprofile applications; application layer developers at UoP and TSI make use of services without knowing details of grid middleware tools and physical resources; service developers can modify implemented (grid) Web services (except at client APIs) at any time without interference to other services/Web applications in most cases.

5.5.3 Performance

The use of today's (grid) Web services techniques for service implementation with layered and componentised services scheme may not be able to offer an efficient grid environment for bioprofile analysis. Web services have certain advantages over other technologies (e.g. CORBA [26] and Java RMI [25]), such as the platform and language independency. They however still have some disadvantages nowadays, such as increased time for finding relevant services/resources, the consumption of more bandwidth for data transfers, and the need for more memory and CPU due to serialisation. This slows down the whole process for dealing with a bioprofile analysis request. The solution to this problem may include the improvement of Web services techniques, the use of other technologies and techniques, and the improvement of design strategy (e.g. finding trade-off between the process efficiency and the ease of collaborative development and expansion of services).

Productivity can be low when employing a number of machines for an analysis job. To investigate the scalability issue, a test was carried out on a Condor pool under PlymGRID [95]. A total number of 1200 tasks were created based on the 400 subjects (each has three EEG records). Four jobs were generated (each with 100, 400, 800 and 1200 tasks, respectively) in order to investigate the performance of the Condor pool. The FD analysis algorithm was used to compute the FD index for each synthetic EEG file. The number of Condor nodes was scaled from 6, 9, 12 up to 50 PCs. Table 5.1 shows the observed execution time for running each specified number of tasks on one machine and at different scales of the Condor pool (nodes from 6 to 50). Figure 5-12 illustrates the speed-up comparison of running different jobs at different scales of the Condor pool. Preliminary results show that the Condor pool with 50 nodes can speed the execution time more than 19 times when compared with the time on a stand-alone PC. It is also observed that running a job with more tasks is relatively more efficient than running a job with fewer tasks. This may be due to the time saving for Condor to create and start a job and the time to transfer executables. This work demonstrates the benefit of EEG analysis using high throughput computing (e.g. by Condor Pool over Grid). However, it can be noticed that the payoff is low when executing the EEG analysis on more than 18 PCs. This may be due to the following reasons:

- Waste of CPU time. The CPU time may be wasted due to the time required to create and start a job, to transfer input files, output files and executables to each execute node/PC, or due to some technical problems of Condor (e.g. preemption);

Table 5.1: Performance evaluation results for scalability (with execution time in minutes and speed up factor compared with 1 PC)

No. of Tasks	100		400		800		1200	
No. of PCs in the Condor pool	exe. time	speed-up	exe. time	speed-up	exe. time	speed-up	exe. time	speed-up
1	64.0	-	255.0	-	511.0	-	765.0	-
6	15.1	4.2	55.3	4.6	103.8	4.9	152.9	5.0
9	11.0	5.8	40.9	6.2	76.3	6.7	113.5	6.7
12	8.7	7.4	32.5	7.9	62.2	8.2	92.1	8.3
15	7.2	8.9	27.1	9.4	52.8	9.7	78.6	9.7
18	6.2	10.3	23.1	11.0	43.3	11.8	63.2	12.1
33	4.9	13.1	17.5	14.6	33.5	15.3	46.9	16.3
50	4.4	14.5	15.0	17.0	28.3	18.1	39.8	19.2

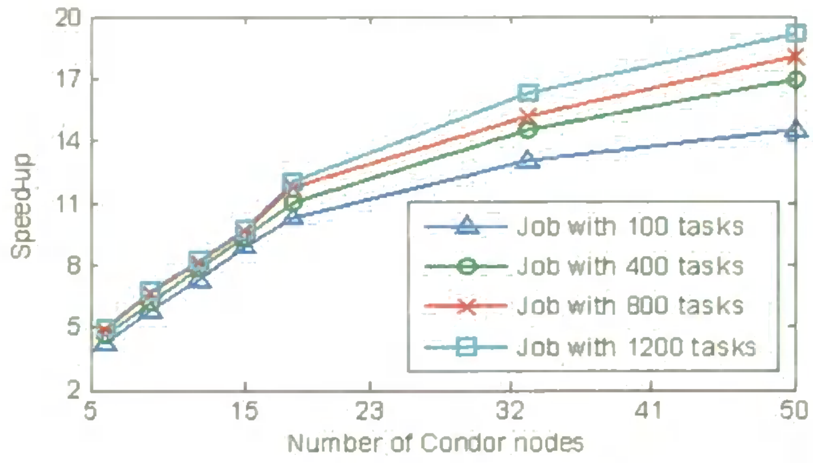


Figure 5.12: Speed-up comparison of running different jobs at different scales of Condor pool

- Lack of checkpoint services. Condor does not provide checkpoint services on Windows machines to date, so that when a running task is interrupted by an unknown incident or other programs on a machine, it requires a restart of the task;
- Processing and networking load of the Condor pool. The running of a Condor job may be interfered by other applications on a processor or under the local network.

5.6 Summary

This chapter has presented the implementation of a distributed EEG analysis environment to support the early detection of dementia. The implementation follows the proposed architecture service model for bioprofile analysis over grid. It was performed in five layers under the bioprofile analysis over grid architecture: from the bottom most databases and compute elements, to the top most user interface. The implementation is based on an EEG analysis scenario, which describes 1) the need for analysis of a patient: Mike, 2) how his EEG data are distributed and 3) how grid may help in carrying out such analysis. Two developed grid test-beds have been presented. The implementation architecture and infrastructure have been illustrated. Implementation details have been given.

Preliminary results show that the implemented grid environment gives supports to clinical users in offering seamless access to distributed resources, and providing clinical evidences (i.e. analysis results) during diagnosis. The results also show that a layered and componentised services development strategy enhances collaborativity and extensibility in developing grid environment for bioprofile analysis. The results reveal the performance issues which must be considered in any future development.

Chapter 6

A Scheme for Knowledge Representation of Grids

6.1 Introduction

In grid, knowledge representation is important in delivering semantics to grid mechanisms for automating information exchange, sharing and integration between grid entities. To date, there are many efforts, which have focused on knowledge representation in some specific sub-domains of grid, such as ontologies for description of (grid) resources [71] [72] [73] and that of (grid) services [74] [75] [76]. However, there is less research that has been made in the aspect of organising multiple descriptions that represent individual grid entities together to represent some parts of, or even an entire, grid environment. This we believe is significant in knowledge-based global coordination between different grid-related components (e.g. resources, applications, security, quality of service and users).

The knowledge representation of a grid environment is not an easy task since the concept of grid covers many computing and communication areas, such as virtualisation, data storage and management, distributed computation, networking, security and QoS. Xing et al [77] proposed a solution to that in 2006, as discussed in Section 2.6.2. However, this solution is not very practical since it makes difficult for developers to use description languages, which are different to the language used by the core ontology, to collaboratively represent individual grid entities and their relationships.

This chapter aims to propose a scheme, which offers a way to organise multiple (heterogeneous) descriptions of individual grid entities together for knowledge representation of grid. This scheme can solve the problem addressed above. It is intended to underpin the concept of bioprofile analysis over grid in the aspect of knowledge-based global coordination between components of bioprofile analysis over a grid.

The chapter is organised as follows. Section 6.2 presents a service-oriented grid model, which further classifies and clarifies grid into 13 components based on the bioprofile analysis over a grid architecture. Section 6.3 proposes a schematic multiplex description architecture for organising heterogeneous descriptions of individual grid entities together for knowledge representation of grid. Section 6.4 proposes a kernel ontology skeleton, which can be used as a framework for developing a kernel ontology, as a part of the description architecture. Section 6.5 demonstrates an implementation example of the description architecture in order to prove the concept. Finally, Section 6.6 concludes the chapter.

6.2 A Service-oriented Grid Model

The term service-oriented grid means a type of grid environment, which makes extensive use of existing and emerging standards from all segments of the Web services community [27]. In a service-oriented grid, services play key roles in the sharing of distributed resources. In order to capture an abstract view of a service-oriented grid, we extend the bioprofile analysis over a grid architecture (see Section 3.4), and consider a service-oriented grid from two points of view - the provisioning and the use of services. Figure 6.1 presents an overview of the proposed service-oriented grid model.

From a provisioning of services point of view, services are interfaces to or representations of virtualised resources (e.g. data, computation, information and knowledge resources). The virtualisations are carried out by grid middleware in a tightly-coupled manner with underlying physical resources (e.g. databases, knowledge-bases and high performance or throughput clusters). Services are held by service containers, which can be regarded as

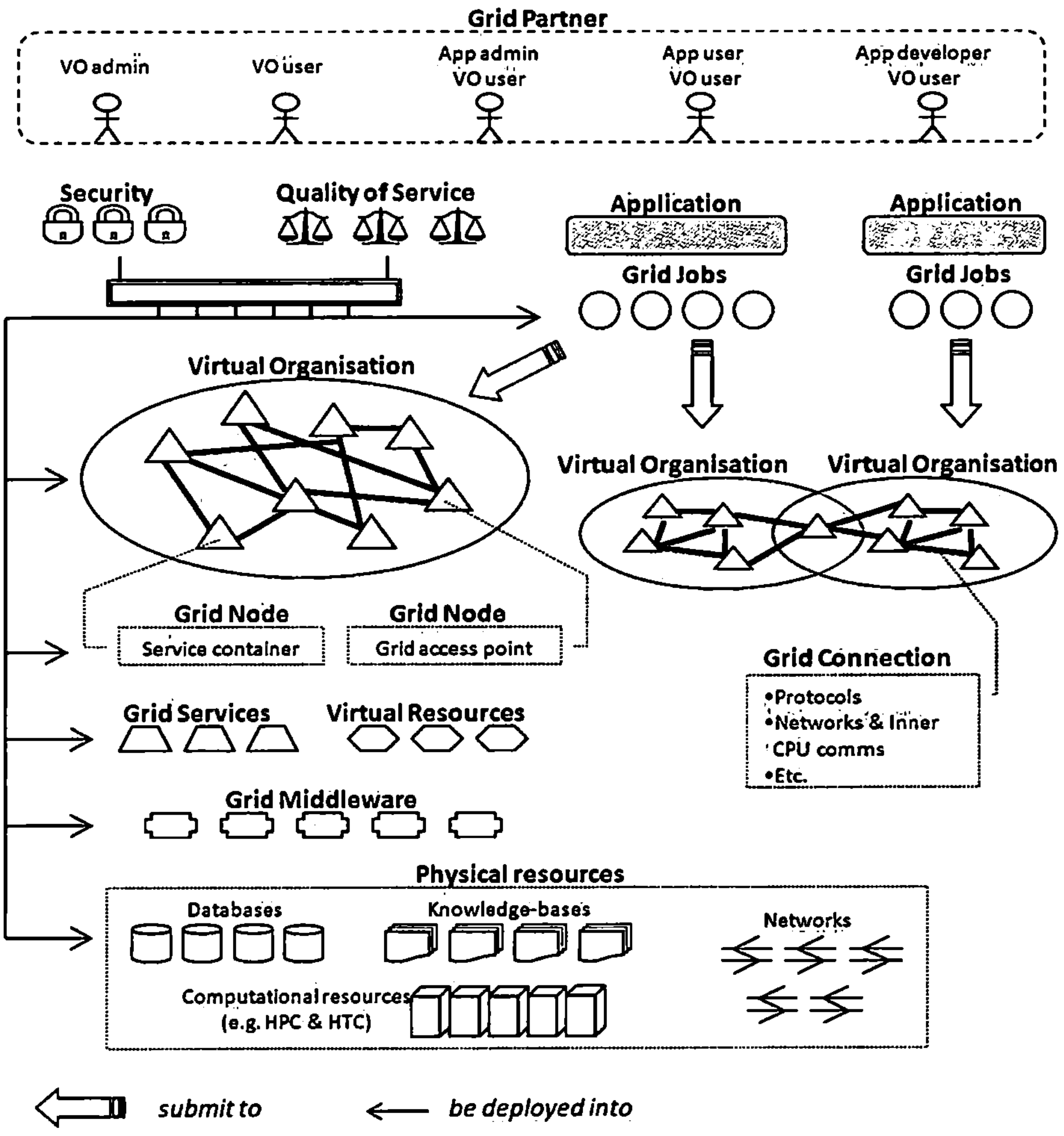


Figure 6.1: A service-oriented grid model

forms of nodes in a grid environment. Another major type of grid node is a grid access point, which can provide interfaces for Virtual Organisation (VO) users or administrators to use or control VOs. A group of grid nodes along with the connections between them can then be organised into a VO for sharing resources within a specific community (Note that a grid node might be held by multiple VOs). In order to provide security functionalities (such as access control) and to deliver the user required Quality of Service (QoS), security and QoS mechanisms and policies can be built into the abovementioned grid entities and/or treated as resources to be accessed from services.

From a use of services point of view, a service-oriented grid is administrated by certain types of VO administrators, and used by different kinds of VO users, who might be the user or the developer of an application. The use of a grid is based on the pre-designed or automatically configured (grid-enabled) applications and via diverse grid jobs to solve specific problems. When security or QoS functionalities provided by a VO do not reach the corresponding requirements of an application (e.g. a VO cannot guarantee a specific level of QoS required by an application), additional security or QoS mechanisms will need to be deployed in the VO.

Note that the interactions between some grid entities described above may have various behaviours. For example, a VO may not have any QoS policies and implementations; and in some cases, a virtual resource can be treated as a part of a grid service. Moreover, in the model, services are only layered logically. That is to say, a group of services can be held by the same type of service containers, but some services may be logically built on top of other services and interact between them.

6.3 A Schematic Multiplex Description Architecture for Knowledge Representation of Grids

The knowledge representation of an entire grid environment is not an easy task since the concept of grid covers many computing and communication areas, such as virtualisation, data storage and management, distributed computation, networking, security and QoS. Furthermore, to present such an environment, we may also face a heterogeneous description environment, since the description of an instance of a defined grid entity can be based on various techniques and/or languages. For example, a grid service can be described using only WSDL [106], or with the support of OWL-S [107]; a simple security management scheme may be described using pure XML, and a more complex one may be described using OWL.

A Schematic Multiplex Description Architecture (SMDA) is proposed to overcome the aforementioned problems, and eventually to support knowledge representation of any grid environment. By the term "multiplex description" here we mean two aspects of practice. One aspect is the separation of concerns. That is to say, the description of the concept of an entire grid environment should be based on separated descriptions of grid sub-domains with distinct features, with their interoperation only relying on additional integration schema/ontologies. Therefore, the knowledge of each defined grid sub-domain can be managed independently. Also, the change of the knowledge in a sub-domain will not lead to a series of changes in other sub-domains.

The other aspect of practice is the use of diverse description techniques and/or languages (e.g. pure XML, RDF and OWL) for grid knowledge representation. This means that the SMDA should be flexible enough to work with different formats and types of description

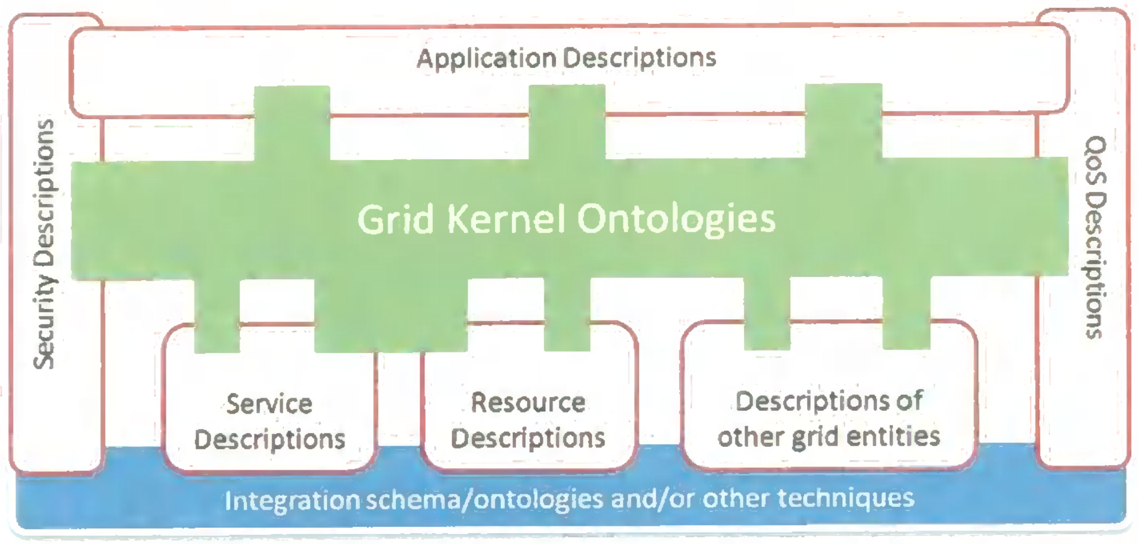


Figure 6.2: SMDA for knowledge representation of grids

of defined grid entities. The integration between them should ideally be loosely-coupled in order to avoid a huge amount of work on attribute and element mapping and description transformations.

Figure 6.2 presents a brief overview of the proposed SMDA for organising description blocks to semantically represent grid environments. The architecture consists of certain categories of descriptions, where:

- Resource Descriptions represent (semantic) descriptions of physical and/or virtual resources, which include the descriptions of resource capabilities, locations, access interfaces, etc. This sort of description normally integrates with security and QoS descriptions in order to provide low-level security and QoS functionalities, such as database access control and cluster monitoring;

- Service Descriptions describe services in grids. A service can be either tightly-coupled or loosely coupled to other services. For those loosely-coupled services, from an OWL-S [107] point of view, the overall ontology structure for service descriptions should contain three major parts, i.e. 1) the service profile for advertising and discovering services; 2) the process model, which gives a detailed description of a service's operation; and 3) the grounding, which provides details on how to interoperate with a service, via messages [107];
- Security Descriptions facilitate security depictions in providing a common grid security vocabulary, and the expression of grid security concepts and relationships between such defined concepts. A well organised group of security descriptions must link the top level of applications' and users' security requests with the bottom level of detailed descriptions of security actions taken in grids;
- QoS Descriptions mainly cover those knowledge representation areas, such as descriptions of user and application QoS requests and the semantics of the monitoring and the management of grid jobs, services and resources, which include ontologies for resource and service matching, job scheduling, etc. Most QoS descriptions work closely with descriptions of other grid entities, such as resources, services and applications;
- Application Descriptions (semantically) describe any computing-related applications. The description of an application may include the roles of the application users and administrators, workflows of the application, what the security and QoS requirements of the application are, etc. The ontology representation of the aforementioned application, the distributed EEG analysis, can be considered as an example of this type of description;

- Grid Kernel Ontologies are the heart of the whole architecture. This kind of (ontology) description links different types of grid sub-domain description together, and defines high-level relationships between them. For example, a grid kernel ontology may bridge specified application and service descriptions to help users and/or systems find out relevant services in grids based on the required functionalities offered by applications (e.g. the analysis of medical images or retrieval of a patient's historical information). Moreover, the relationships between the grid kernel ontologies and the above-described types of descriptions should, in our opinion, be loosely-coupled. That is to say, in a grid kernel ontology, we may only define the description instance and related description schema/ontology of a grid entity (e.g. service, resource, QoS and security) instead of describing the grid entity in detail to maximally reduce the work of interoperation between the grid kernel ontology and other grid sub-domain descriptions (e.g. the mapping of different terms and semantically bridging different concepts);
- Integration schema/ontologies provide (semantic) additional integration functionalities between descriptions of different grid sub-domains when a single grid kernel ontology is unable (e.g. it has unavoidable mappings between descriptions) or inappropriate (e.g. low level interoperation between descriptions) to deliver. An example of this kind of description can be the mappings between the functions a VO can offer with application requirements in sub-domains like security, QoS, service, partners, etc.;
- Other grid-related descriptions mainly include descriptions of other grid entities, such as the entity grid job, which may define semantics for job description formats, relationships between grid jobs and application workflows and between grid jobs and services, etc.

6.4 A Kernel Ontology Skeleton

A grid kernel ontology skeleton is proposed to provide a framework for developing grid kernel ontologies. It aims to define classes of main grid entities, fundamental relationships between those classes and basic constraints to those classes, and means for the integration of grid kernel ontologies with grid sub-domain descriptions. The design of the grid kernel ontology skeleton is based on the proposed service-oriented grid model, as described in Section 6.2. The ontology skeleton can be easily extended by adding additional classes, properties and constraints. One important feature of the ontology skeleton is that it uses a loosely-coupled manner for the integration with grid sub-domain descriptions. The feature is expected to offer componentised development of descriptions in different grid domains and to maximally reduce the interoperation work between those descriptions. It can thus adapt to the rapid development of grid technology.

In our grid kernel ontology skeleton development, the Web Ontology Language (OWL) [65], as a well known W3C recommendation, is used for the implementation. The selection of OWL provides more facilities for expressing meaning and semantics than XML, RDF, and RDF-S, and makes the ontology easier to extend. Moreover, in the implementation, Protege [108] is used as the ontology editor. OWLViz [109] is employed for ontology visualisation. RacerPro [110] is used as the inference system. The latest version of the implemented grid kernel ontology skeleton is downloadable at [111].

6.4.1 Class Definition and Hierarchy

In the proposed service-oriented grid model, we described a set of grid entities. The two top-layer entities are VO and application. Others are the associated components, which can be constructed to support them. For the design of the grid kernel ontology skeleton, we map all the described grid entities into main abstract classes and their relationships into properties and constraints. Table 6.1 presents the main (abstract) classes and their descriptions as defined in the ontology skeleton. Some lowerlevel sub-classes are shown in Figure 6.3.

The ontology skeleton uses subclasses, as "Named***" (e.g. NamedService) / "***Instance" (e.g. GridConnectionInstance), "***Description" (e.g. ApplicationDescription) and "***DescriptionSchema" (e.g. JobDescriptionSchema) to provide information of grid sub-domain descriptions instead of directly defining specific grid sub-domain properties and/or constraints to main grid entities to avoid, such as attribute overlapping with grid sub-domain descriptions. For example, the class "Service" contains two subclasses, NamedService and ServiceDescription. Each named service can be described by one or more service description(s), and each service description is accompanied with a single or a set of description schemas/ontologies. Thus, the functions offered by grid kernel ontologies and service descriptions can be distinguished. A grid kernel ontology is then able to work with various types and formats of service descriptions. The technical change of service descriptions will thus not lead to intensive mappings between them. In other words, a user or a system will be able to find a required service in a grid kernel ontology, but will obtain the detailed descriptions of the various implementations of the service from service ontologies or other knowledge representation techniques based on the instances of ServiceDescription and

Table 6.1: Main/Abstract classes of the proposed grid kernel ontology skeleton

Class	Description
PhysicalResource	Any computing-related resources, such as a PC, a high performance computer, a binary file, a data storage and communication networks.
GridMiddleware	Software stacks designed to virtualise and provide access to physical resources.
VirtualResource	Logical representation of a/multiple physical resource(s).
Security	The control of risks related to the access of a VO, a grid service, data, etc.
QoS	Quality measure and control in order to provide different priority to different grid users, or to guarantee a certain level of performance to grid applications.
Service (Grid Service)	Software components, which provide platform-independent protocols and standards used for exchanging data between applications. They can be either stateful or stateless, as compared with a typical Web service.
GridJob	Paticular units of work to solve a/multiple problem(s) defined by a grid application. They can be described either abstractly or concretely.
GridNode	Nodes in a grid environment, including service containers, grid clients, etc.
GridConnection	Logical connections between grid nodes.
Partner	A specified role in the access to resources and/or services within a VO or a grid application.
GridVO	A type of administrative domain for sharing resources across different institutions and/or individuals in order to achieve a specific goal.
Application	A collection of work items that can carry out complex computing tasks by using grid services and resources.
Support	Additional entities which provide global support for descriptions of grid components.

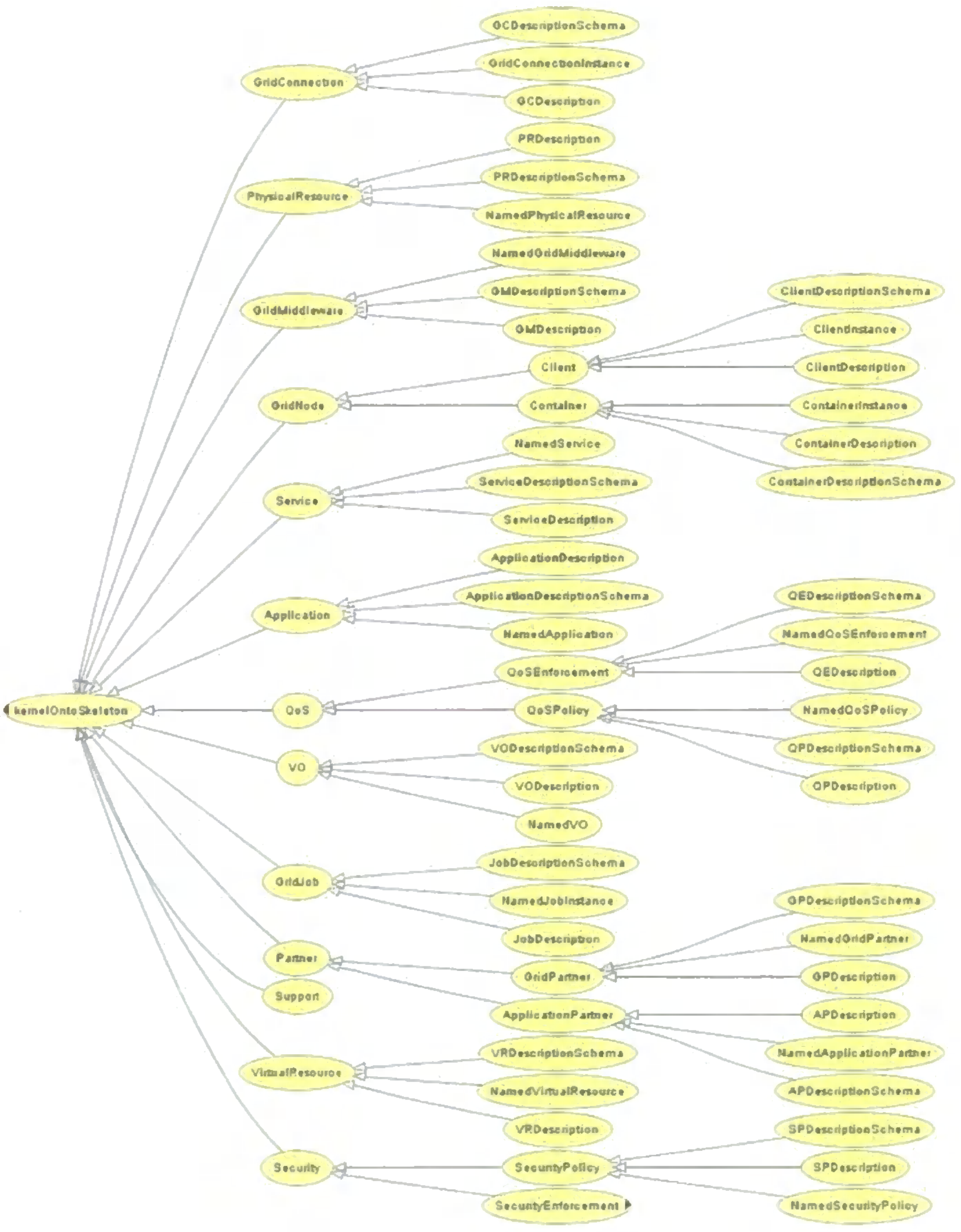


Figure 6.3: Class hierarchy of the grid kernel ontology skeleton

ServiceDescriptionSchema. When technical changes of service descriptions happen, we only need to add or modify the instances of the above two subclasses in a grid kernel ontology. This avoids a huge amount of work on attribute and element mapping and description transformation.

Moreover, we divided "partners" into two categories, and defined two (sub)classes, GridPartner and ApplicationPartner, under the class Partner. The GridPartner class is for the descriptions of individual roles for using and managing grid facilities. The ApplicationPartner class is for descriptions of the individual roles for using and managing applications. For instance, for bioprofiling, we may have certain categories of users and administrators, such as patient, researcher, administrator and clinician.

The proposed grid kernel ontology skeleton also contains two important (sub)classes: QoSPolicy and SecurityPolicy, which are used to hold all "policy" instances for other grid entities to invoke. For example, a NamedVO instance may have a NamedSecurityPolicy instance for access control; and a NamedApplication instance may require a NamedQoSPolicy instance for the statement of its QoS requirements.

6.4.2 Properties and Constraints

To represent relationships between and constraints to the determined classes, we have also defined a set of properties and constraints.

The defined properties include:

- "hasDescription", which describes an one way "has description" relationship between the object class "Named***" or "***Instance" and the subject class "***Description";
- "useDescriptionSchema", which describes an one way "use description schema" relationship between the object class "***Description" and the subject class "***DescriptionSchema";
- "hasPartner", which describes an one way "has partner (i.e. user, administrator, etc.)" relationship between, such as the object class "NamedApplication" and the subject class "NamedApplicationPartner", and the object class "NamedVO" and the subject class "NamedGridPartner";
- "requireService", which describes an one way "require service" relationship between, the object class, such as "NamedApplication", and the subject class "NamedService";
- "hasService", which describes an one way "has service" relationship between the object class, such as "NamedVO", and the subject class "NamedService";
- "hasGridMiddleware", which describes an one way "has grid middleware" relationship between the object class, such as "NamedVO" and the subject class "NamedGridMiddleware";
- "supportApplication", which describes an one way "support application" relationship between the object class, such as "NamedVO", and the subject class "NamedApplication";
- "achieveByJob", which describe an one way "achieved by job" relationship between the object class "NamedApplication" and the subject class "NamedGridJob";

- "requireVirtualResource", which describes an one way "require virtual resource" relationship between the object class, such as "NamedApplication" and the subject class "NamedVirtualResource";
- "hasVirtualResource", which describes an one way "has virtual resource" relationship between the object class, such as "NamedVO" and the subject class "NamedVirtual-Resource";
- "hasPhysicalResource", which describes an one way "has physical resource" relationship between the object class "NamedVirtualResource" and "NamedPhysicalResource";
- "hasContainer", which describes an one way "has (service) container" relationship between the object class "NamedVO" and the subject class "ContainerInstance";
- "hasClient", which describes an one way "has (grid) client" relationship between the object class "NamedVO" and the subject class "ClientInstance";
- "implements", which describes an one way "is the implementation of" relationship between, such as the object class "NamedSecurityEnforcement" and the subject class "NamedSecurityPolicy";
- "connects", which describes an one way, but reversible "has communicatory connection with" relationship between the object class "ContainerInstance" and the subject class "ClientInstance" or vice versa;
- "hasGridConnection", which describes an one way "has communicatory connections between" relationship between the object class, such as "NamedVO" and the subject class "GridConnectionInstance";

- "isA", which describes an one way "is a" relationship between, such as the object class "NamedApplicationPartner" and the subject class "NamedGridPartner" (e.g. a bioprofile analysis application partner: clinician, is a normal grid user);
- "hasSecurityPolicy", which describes an one way "has security policy" relationship between the object class, such as "NamedApplication", and the subject class "NamedSecurityPolicy";
- "hasSecurityEnforcement", which describes an one way "has security enforcement" relationship between the object class, such as "NamedVO" and the subject class "NamedSecurityEnforcement";
- "hasQoSPolicy", which describes an one way "has QoS policy" relationship between the object class, such as "NamedVO" and the subject class "NamedQoSPolicy";
- "hasQoSEnforcement", which describes an one way "has QoS enforcement" relationship between the object class, such as "NamedVO", and the subject class "NamedQoSEnforcement";

General constraints used in the design include:

- "owl:allValuesFrom", which describes, for example, a NamedService instance that can only have description instance(s) from the class ServiceDescription;
- "owl:minCardinality", which describes, for example, a ServiceDescription instance that must have at least one ServiceDescriptionSchema (so the indicated service description can be automatically parsed by the support of a or a group of predefined schemas).

Other constraints are defined specific to individual classes. Those constraints indicate the relationships between a specific class with other classes. For example, for an application, we may need to consider what its description is; what kinds of people can use this application; what security and QoS policy it has defined; what services are required to deliver specific functionalities (e.g. data query and computation) defined by the application; what virtual resources are needed in order to provide facilities required by the application; and how to use grid jobs to solve various problems defined by the application. So, for the NamedApplication class, we may have constraints, such as the following:

- hasDescription only (i.e. owl:allValuesFrom) ApplicationDescription;
- hasPartner only NamedApplicationPartner;
- hasQoSPolicy only NamedQoSPolicy;
- hasSecurityPolicy only NamedSecurityPolicy;
- requireService only NamedService;
- archieveByJob only NamedJobInstance;
- requireVirtualResource only NamedVirtualResource.

6.5 An Implementation of the Schematic Multiplex Description Architecture

An example of the implementation of SMDA is performed in order to demonstrate the SMDA approach in utilising existing description techniques and organising heterogeneous descriptions of grid entities to support the knowledge-based global coordination between different grid components. The implementation example is based on the implemented grid-enabled application: EEG analysis over grid for early detection of dementia, as described in Chapter 5. The example considers related descriptions in three major grid entities (i.e. application, partner and service) and a grid kernel ontology.

6.5.1 An Implementation of SMDA-based Descriptions for EEG Analysis over Grid for Early Detection of Dementia

A. The Description of an Application

A description of the application, EEG analysis for early detection of dementia, is implemented based (partly) on the generic application ontology developed by the K-WF Grid project [75]. The application description semantically describes certain important entities of an application, including name, version, text-based description, involved functions (e.g. Fractal dimension and Tsallis entropy analysis), input, and output of an application. Appendix D shows an example of the implementation using the ontology language OWL.

B. The Description of a Partner

The descriptions of two types of partners: grid and application partners, are implemented. The grid partner descriptions are implemented based on the person (XML) schema developed by IeSE [112]. The important entities concerned in such descriptions include personal information (e.g. person name, title and contact details), a person's role, and a person's ability in a grid environment. An example of such a description implemented in plain XML is presented in Appendix E.1.

The application partner descriptions are implemented with the consultation of the person schema of the draft version of the bioprofiling data model [88]. Such descriptions involve four types of application partners, as clinician, patient, researcher and bioprofiling administrator. The important entities concerned include personal information, a person's role and a person's ability in the EEG analysis application. An example of a clinician's description implemented in plain XML is shown in Appendix E.2.

C. The Description of a Service

For simplicity, the WSDL files used in the implementation of those services described in Section 5.4.6 are directly used as the descriptions of services. An example of an EEG analysis service is presented in Appendix F.

D. A Grid Kernel Ontology

A grid kernel ontology has been implemented based on the grid kernel ontology skeleton. All the above introduced descriptions were registered as instances into the grid kernel ontology, in terms of the name or ID of a grid entity instance, its description name(s)/ID(s) and corresponding description schema(s) used. For example, for an EEG analysis service, we register the name or ID of the service, EEGAnalysis as a service instance into the NamedService class, one of its descriptions, EEGAnalysis.wsdl, as a service description instance into the ServiceDescription class, and the description schema WSDL as a service description schema instance into the ServiceDescriptionSchema class. The ontology also directly uses those properties and constraints defined by the ontology skeleton to express the relationships between those instances. For example, the use of the property requireService and related constraints to the class NamedApplication to express the implemented EEG analysis application requires the support from the EEG analysis service and the subject information query service. Appendix G shows an example of the implementation of the grid kernel ontology using the ontology language OWL.

6.5.2 Evaluation of the Implementation

The above-described implementation has been evaluated in the aspects of the delivery of knowledge-related functionalities in the support of global coordination between grid components.

A. Find required description(s)

This is important in retrieving required descriptions of grid entity instances in order to let either a human or a machine to understand, for example, what an instance is about, its requirements and the functionalities it can deliver, so as to support the use and management of appropriate grid components to meet specific requirements defined by an application.

This functionality is achievable by querying the grid kernel ontology based on the name/ID of an instance of one of those defined Named*** or ***Instance classes. The grid kernel ontology will then provide the names/IDs of all registered descriptions of the instance and corresponding schemas used in formatting those descriptions, respectively. The name(s)/ID(s) of the selected description(s) can then be used as abstract information to locate (e.g. by querying index services) and retrieve the selected description(s). The name(s)/ID(s) of related schemas can be used as indication information to support the selection of the correct parser(s) for parsing those description(s).

B. Interoperability between different grid entities

The implementation example also delivers certain functionalities for the interoperability between different grid entities, including

- Finding appropriate services required by an application or a service by querying the NamedService class based on the name/ID of the application and the property requireService. This functionality is important in service composition;
- Finding application partner groups corresponding to an application by querying the NamedApplicationPartner class based on the name/ID of the application and the property hasPartner. This functionality is important in the understanding

of partner roles in an application. It is expected to clarify application security and QoS policies for different types of application partners;

- Mapping application partner groups to grid partner groups by querying the NamedGridPartner class based on the name/ID of an application partner group and the property isA. This functionality is important in the understanding of the relationship between application partner groups and grid partner groups. It is expected to map security and QoS policies for a specific application user group to those for a grid user group;

6.6 Summary

This chapter has presented a scheme for the knowledge representation of grids. The scheme mainly consists of a Schematic Multiplex Description Architecture (SMDA) and a grid kernel ontology skeleton. The description architecture can be used to organise heterogeneous descriptions of individual grid entities together for the knowledge representation of grids. The grid kernel ontology skeleton, which has been developed based on the proposed service-oriented grid model, can be used as a framework for developing a kernel ontology, as a part of the description architecture. In this chapter, an implementation example of the description archtecture has also been presented in order to demonstrate the architecture's approach in utilising existing description techniques and organising heterogeneous descriptions of grid entities to support the knowledge-based global coordination between different grid components.

Overall, the scheme can solve the problem of organising heterogeneous descriptions of individual grid entities together for knowledge representation of grid environments either partly or entirely. The scheme underpins the concept of bioprofile analysis over grid in the aspect of the knowledge-based global coordination between components of bioprofile analysis over grid. The scheme is generic, so is also applicable to other sorts of grid environments.

Chapter 7

Discussion, Suggestion for Future Work and Conclusion

7.1 Introduction

Existing ICT technologies are inappropriate for bioprofiling because of the difficulties in the use and management of heterogeneous IT resources at different bioprofile centres. Grid as an emerging resource sharing concept fulfils the needs of bioprofiling in several aspects, including the discovery, access, monitoring and allocation of distributed bioprofile databases, computation resources, bioprofile knowledge bases, etc. However, the challenge of how to integrate the grid and bioprofile technologies together in order to offer an advanced distributed bioprofile environment to support individualized healthcare remains.

The aim of this project was to develop a framework for bioprofile analysis over grid to support individualised healthcare. The framework provides a basis for a "grid-based" solution to the challenge of "distributed bioprofile analysis" in bioprofiling.

This chapter discusses the main contributions of this work, its limitations and possible future work.

7.2 Contributions to Knowledge

In summary, this thesis presents research that has achieved the following:

A. An Architecture for Bioprofile Analysis over Grid

The design of a suitable architecture is fundamental to the development of bioprofile analysis over grid. The investigation of an architecture for bioprofile analysis over grid was based on those questions addressed in "Research Question A". An investigation was undertaken into aspects of the general concept and characteristics of bioprofiling, bioprofile analysis requirements, the main distributed bioprofile analysis system/environment components and their relationships, and the overall structure of bioprofile analysis over grid. The developed architecture provides an understanding of bioprofile analysis over grid, and also creates a means for the categorisation, determination and organisation of core grid components in order to support the development and use of grid for bioprofile analysis. The developed architecture delivers a possible answer to the second research hypothesis.

B. A Service Model for Bioprofile Analysis over Grid

Service is a core grid component. It normally provides platform independent protocols and standards used for exchanging data between clients and resources. Our investigation of a service model for bioprofile analysis over grid is based on those questions addressed in "Research Question B". The investigation focused on the issues of service collaboration, extensibility and reusability. It also used EEG analysis as an example of bioprofile analysis to address the utilisation of existing services and the development of new services in order to meet requirements defined by bioprofile analysis applications. The investigation has delivered a service design principle, a service architecture for bioprofile analysis over grid, and an EEG analysis service model. The developed service model provides the knowledge in construction and organisation of services in support to distributed bioprofile analysis applications.

C. Two grid test-beds and a practical implementation of EEG analysis over grid.

The investigation of grid test-bed and grid environment for EEG analysis was based on those questions addressed in "Research Question C". Grid test-beds are fundamental to any practise-required grid-related projects. The development of the BIOPATTERN grid has provided practical knowledge in utilising existing bioprofile, grid and other ICT technologies to provide an essential experimental platform for research in bioprofiling over grid. The building of PlymGRID offers knowledge in the development of a campus grid environment. The implementation of an EEG analysis over grid demonstrates how resources, grid middleware and services can be utilised, organized and implemented to support EEG analysis for the early detection of dementia. The result of implementation proves the third research hypothesis that existing services and/or tools can be directly utilised or organised to support distributed bioprofile analysis. The evaluation results show that the implemented grid-enabled EEG analysis platform delivers support to clinical users in offering seamless access to distributed resources, and providing clinical evidence (in terms of biomarkers) during diagnosis. This partly proves the first research hypothesis that grid can provide solutions for distributed bioprofile analysis. The results also show that a layered and componentised service development strategy enhances collaboration and extensibility in developing a grid environment for bioprofiling. This proves the fourth research hypothesis that the issues of "application-driven development", "separation of concerns", "layered services" and "generic" are important in the successful design of a grid service. The developed distributed EEG analysis environment can be used to support a variety of research activities in EEG analysis.

D. A Scheme for Knowledge Representation of Grids

Knowledge support is important in knowledge-based global coordination between different grid components for any grid environments. However, hitherto only limited work has been done in utilising and organising various knowledge representation languages and schemas/ontologies to provide collaborative knowledge representation of grids. The investigation of a knowledge representation scheme has thus been undertaken based on those questions addressed in "Research Question D". The investigation has addressed the determination of major entities in representing the bioprofile analysis over grid concept. It has delivered a flexible and extensible way to organize multiple (heterogeneous) descriptions of individual grid entities together for the knowledge representation of grids. It has also demonstrated an approach in utilising existing description techniques and organising heterogeneous descriptions of grid entities to support the knowledge representation of grids. The proposed scheme is generic and so underpins the concept of bioprofile analysis over grid in the aspects of knowledge-based global coordination between grid components. The scheme offers a possible answer to the fifth research hypothesis that a group of multiple (heterogeneous) descriptions can be organised to describe a grid environment.

7.3 Limitations of Current Work and Discussion

There are certain limitations in the project which should be addressed:

A. Limited Development in Bioprofile, Grid and Knowledge Representation Technologies

As discussed in chapter 2 and 3, at present, none of bioprofile, grid and knowledge representation is mature technology. Their concepts will obviously be extended. Related new techniques will also be developed in the future. This could lead to knowledge and development gaps in this work, especially on the issues of unforeseeable service types, which need to be defined in the proposed service model; under-developing bioprofile, grid and other ICT techniques, which need to be further clarified before utilisation; new healthcare-related legislations, which might change requirements of bioprofile in the aspects of privacy and security; etc.

B. Limited research resources

The practical part of this work is based on a number of bioprofile, knowledge representation, grid middleware, and other ICT resources. However, there is far more to representing a distributed bioprofile analysis environment in reality, especially in the aspects of the use of real bioprofile data, the access of real clinical databases and the limited scale of the grid test-beds.

C. Limited Demonstration and Validation of the Work

This thesis has presented a framework of bioprofile analysis over grid. Some demonstrations and validations for the work have been carried out, however the limited development in bioprofile, grid and knowledge representation technologies and limited research resources restrict those demonstrations and validations further in every aspect. For example, the implementation of distributed EEG analysis platform only considers three analysis algorithms due to limited numbers of software implementations of EEG analysis algorithms. There could be unseen issues (e.g. special requirements on privacy and confidentiality in distributed environments) when new analysis algorithms introduced into the grid; and there are absences of QoS enforcement and weak security support from those basic security mechanisms in the EEG analysis platform due to the lack of QoS and security contributions from the grid community.

7.4 Suggestion for Future Work

This project only delivers a foundation for the development of bioprofile analysis over grid. In the future, the bioprofile analysis over grid research could lead to a number of topics, which mainly include:

A. The Extension of the Service Model for Bioprofile Analysis over Grid

   In the future, there will be plenty of new types of grid services coming out, such as services for workflow management, and automatic security and QoS enforcements based on predefined policies. It is necessary to address what roles those types of services have and how to utilise them to support bioprofile analysis over grid.

B. The Investigation and Development of Bioprofile Analysis Applications and Associated Services

There are a number of bioprofile analysis applications, such as the analysis and fusion of medical images, genomic data and electrophysiological data. Each of those applications has different characteristics and requirements. Therefore, it is essential to investigate each user-required bioprofile analysis application, and to utilise as well as to develop associated services to support those applications.

C. The Investigation of Bioprofiling Facilities

Bioprofiling requires many types of facilities, such as biomedical equipments, analysis software/tools, and bioprofile data management software/tools. The investigation of bioprofiling facilities is important in providing characteristic and interface information of such facilities in order to build additional grid-related mechanisms for virtualisation and interface transformation for sharing.

D. The Utilisation of Tomorrow's Grid Middleware and Services

This could relate to the future work of the service model extension. But it also needs to improve and extend existing grid functionalities by updating and upgrading existing grid middleware and services to advanced ones when it is necessary.

E. The Development of a Bioprofiling Data Model

The success of bioprofile analysis over grid is inseparable from a bioprofiling data model. The development of such a model is essential in standardising federated data access interfaces, the descriptions of bioprofile data and facilities, security and QoS policies, etc.

F. The Improvement of the Knowledge Representation Scheme

The proposed knowledge representation scheme is based on today's knowledge of grid and knowledge representation. The scheme is flexible for any type of description techniques, some parts of the scheme however still need to be improved in the future. For example, it is necessary to extend the schematic multiplex description architecture and the grid kernel ontology skeleton when new concepts of grid come out.

G. The Knowledge Representation of Grids

As discussed in Chapter 6, the knowledge representation of grids is important in delivering knowledge support for global coordination between grid components. To represent a grid environment based on the proposed knowledge representation scheme, it is necessary to carry out two parts of work. One is to utilise Grid Sub-domain Description Schemas/Ontologies. The other is to develop a grid kernel ontology.

There are a number of grid sub-domain description schemas/ontologies, which are under development or will be developed, such as the introduced the service ontology OWL-S [107] and those resource ontologies [71] [72] [73], and future security and QoS related schemas/ontologies. They are utilisable in descriptions of grid sub-domain entities.

A Grid Kernel ontology is essential in the knowledge representation scheme in order to integrate descriptions of different grid entities together to support the knowledge representation of grids. The development of a grid kernel ontology can rely on the developed grid kernel ontology skeleton and should be based on a specified grid environment.

H. The Utilisation of Knowledge Access Middleware and Services

There are some distributed knowledge access related middleware and services that are under development, such as OGSADAI-RDF [113]. Those middleware and services can provide appropriate support in the realisation of the knowledge representation scheme in aspects of the access of distributed descriptions of grid entities. The utilisation is a solution specific to the problem of knowledge representation of grids, but can also be considered as a part of the future work of the "utilisation of tomorrow's grid Middleware and services".

7.5 Conclusion

An important conclusion of this project is that it delivers a framework for bioprofile analysis over grid. This provides a basis for a "grid-based" solution to the challenge of "distributed bioprofile analysis" in bioprofiling. The development of the framework covers four important areas of research, including an architecture for bioprofile analysis over grid, a service model for bioprofile analysis over grid, a demonstrative implementation of grid environment for EEG analysis, and a knowledge representation scheme in support of global coordination between components of bioprofile analysis over grid. The outcomes of this research have been presented in this thesis. They have provided an understanding of the bioprofile analysis over grid concept, and standardisations as well as approaches for building a grid environment for bioprofile analysis.

Based on the results obtained from our design and implementation, we conclude as follows:

- Grid, as an emerging resource sharing concept can deliver the decentralised management and seamless access of distributed heterogeneous bioprofile analysis resources. It can enhance the efficiency of healthcare.
- The existing grid services and/or middleware tools can be utilised to support the implementation of grid environment for bioprofile analysis.
- The features of "application-driven development", "separation of concerns", "layered services" and "generic" can enhance the collaborative nature of the grid service development and deliver extensibility and reusability to services.
- The proposed Schematic Multiplex Description Architecture can be a solution to the use of multiple heterogeneous descriptions of grid entities for the knowledge representation of grids.
- At present, bioprofile, grid and knowledge representation are not mature technologies. The concept of bioprofile analysis over grid must be improved and/or extended with the development of such technologies.

The outcomes of this research may be appropriate for other healthgrid and semantic grid related projects. The proposed architecture and service model for bioprofile analysis over grid can be referenced to the building of other healthgrid environments since most parts of the architecture and the service model are generic to different healthgrids. The implemented grid test-beds can be used or extended to support other types of distributed data analysis. The proposed Schematic Multiplex Description Architecture is generic. It can thus be referenced to other work on the development of semantic grids. The developed grid kernel ontology can be used as a framework for the design and development of grid-related ontologies.

The building of a grid environment for bioprofile analysis is a long-term, step-by-step process. In the future, developments could focus on the investigation of real bioprofiling facilities; the utilisation of future grid middleware and services; and the building of bioprofiling data model, as addressed in the "Suggestion for Future Work" section. The development must follow users' needs, and be driven by users' feedbacks in order to deliver user-required bioprofile analysis functionalities to clinicians and bioprofile researchers via secure, reliable, seamless and efficient grid environments.

References

[1] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. Von Reich, "The Open Grid Services Architecture (OGSA), version 1.5." GFD-I.080, July 2006.

[2] S. R. Amendolia, F. Estrella, W. Hassan, T. Hauer, D. Manset, R. McClatchey, D. Rogulinl, and T. Solomonides, "Mammogrid: A service oriented architecture based medical grid application," in *Proceedings of the 3rd International Conference on Grid and Cooperative Computing*, pp. 939–942, 2004.

[3] S. Santini and A. Gupta, "The role of internet imaging in the biomedical informatics research network.," in *Proceedings of SPIE*, p. 5018, 2003.

[4] "Health-e-Child, a integrated platform for european paediatrics based on a grid-enabled network of leading clinical centres." http://www.health-e-child.org/.

[5] S. R. Amendolia, F. Estrella, C. D. Frate, J. Galvez, W. Hassan, T. Hauer, D. Manset, R. McClatchey, M. Odeh, D. Rogulin, T. Solomonides, and R. Warren, "Development of a grid-based medical imaging application," in *Proceedings of Healthgrid 2005, from Grid to Healthgrid*, pp. 59–69, 2005.

[6] S. Lloyd, M. Jirotka, A. C. Simpson, R. P. Highnam, D. J. Gavaghan, D. Watson, and J. M. Brady, "Digital mammography: a world without film?," *Methods of Information in Medicine*, vol. 44, no. 2, pp. 168–169, 2005.

[7] J. S. Grethe, C. Baru, A. Gupta, M. James, B. Ludaescher, M. E. Martone, P. M. Papadopoulos, S. T. Peltier, A. Rajasekar, and S. Santini, "Biomedical informatics research network: Building a national collaboratory to hasten the derivation of new understanding and treatment of disease," in *Proceedings of Healthgrid 2005, from Grid to Healthgrid*, pp. 100–109, 2005.

[8] "Biogrid, construction of a supercomputer network." http://www.biogrid.jp/.

[9] BIOPATTERN, "European network of excellence - computational intelligence for biopattern analysis in support of ehealthcare (FP6-2002-IST-1 No. 508803)." 2004-2008, www.biopattern.org.

[10] MySQL, "The worlds most popular open source database." http://www.mysql.com/.

[11] PostgreSQL, "The worlds most advanced open source database." http://www.postgresql.org/.

[12] "Apache Xindice." http://xml.apache.org/xindice/.

[13] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, 2001.

[14] L. Sun, P. Hu, C. Goh, B. Hamadicharef, M. Hess, E. C. Ifeachor, I. Barbounakis, M. Zervakis, N. Nurminen, and A. Varri, "Bioprofiling over grid for early detection of dementia," in *Proceedings of the First International Conference on Scalable Information Systems (INFOSCALE 2006)*, (Hong Kong), May 2006.

[15] L. Sun, P. Hu, C. Goh, B. Hamadicharef, E. C. Ifeachor, I. Barbounakis, M. Zervakis, N. Nurminen, A. Varri, R. Fontanelli, S. Di Bona, S. Guerri, S. La Manna, K. Cerbioni, E. Palanca, and A. Starita, "Bioprofiling over grid for ehealthcare," in *Proceedings of the HealthGRID 2006*, (Valencia, Spain), June 2006.

[16] P. Hu, L. Sun, C. Goh, B. Hamadicharef, E. C. Ifeachor, I. Barbounakis, M. Zervakis, N. Nurminen, A. Varri, R. Fontanelli, S. Di Bona, S. Guerri, S. La Manna, K. Cerbioni, E. Palanca, and A. Starita, "The biopattern grid: Implementation and applications," in *Proceedings of the 5th All Hands Meeting 2006*, (Nottingham, UK), Sept. 2006.

[17] P. Hu, L. Sun, and E. C. Ifeachor, "A framework for bioprofile analysis over grid." Submitted to the IEEE System Journal.

[18] P. Hu, A. Anastasiou, L. Sun, and E. C. Ifeachor, "A model for bioprofile over grid in support of ehealthcare," in *Proceeding of the 3rd International Conference on Computational Intelligence in Medicine and Healthcare (CIMED2007)*, (Plymouth, UK), July 2007.

[19] P. Hu, L. Sun, and E. C. Ifeachor, "An approach to structured knowledge representation of service-oriented grids," in *Proceedings of UK e-Science Programme All Hands Meeting 2007*, (Nottingham, UK), Sept. 2007.

[20] "What is middleware." http://www.middleware.org/whatis.html.

[21] D. Bakken, "middleware." Washington State University.

[22] K. Liakos, A. Burger, and R. Baldock, "Distributed processing of large biomedical 3d images." Lecture Notes in Computer Science 3402, 2005.

[23] J. Chun and J. Son, "A corba-based telemedicine system for medical image analysis and modelling." 14th IEEE Symposium on Computer-Based Medical Systems (CMBS'01), 2001.

[24] A. Page, T. Keane, and T. Naughton, "Bioinformatics on a heterogeneous java distributed system." 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), 2005.

[25] "Java Remote Method Invocation (Java-RMI)." http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp.

[26] "CORBA: Common Object Request Broker Architecture." http://www.omg.org/.

[27] L. Srinivasan and J. Treadwell, "An overview of service-oriented architecture, web services and grid computing." HP Software Global Business Unit, Nov. 2005.

[28] I. Foster, A. Roy, and V. Sander, "The grid: Blueprint for a new computing infrastructure." Morgan Kaufmann, 1999.

[29] I. Foster and A. Tamnitchi, "On death, taxes, and the convergence of peer-to-peer and grid computing." 2nd International Workshop on Peer-to-Peer System (IPTPS03), Feb. 2003.

[30] "Enabling Grids for E-sciencE (EGEE)." http://www.eu-egee.org/.

[31] "The CoreGRID network of excellence." http://www.coregrid.net/.

[32] "NextGRID: Architecture for next generation grids." http://www.nextgrid.org/.

[33] "Java enterprise platform, Java EE/J2EE." http://java.sun.com/javaee/.

[34] "WebSphere, integration and application infrastructure software." http://www-306.ibm.com/software/websphere/.

[35] "The .NET framework."
http://msdn2.microsoft.com/en-us/netframework/default.aspx.

[36] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe, "The WS-Resource Framework (WSRF), version 1.0." White paper, Mar. 2004.

[37] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, P. Vanderbilt, and D. Snelling, "Open Grid Services Infrastructure (OGSI) version 1.0." Global Grid Forum, GWD-R, June 2003.

[38] I. Foster, "Globus toolkit version 4: Software for service-oriented systems," in *Proceedings of IFIP International Conference on Network and Parallel Computing*, pp. 2–13, 2005.

[39] B. Sotomayor, "The Globus toolkit 4 programmers tutorial." University of Chicago, 2005.

[40] "Oracle Database." http://www.oracle.com/database/.

[41] M. Antonioletti, M. P. Atkinson, R. Baxter, A. Borley, N. P. Chue Hong, B. Collins, N. Hardman, A. Hume, A. Knox, M. Jackson, A. Krause, S. Laws, J. Magowan, N. W. Paton, D. Pearson, T. Sugden, P. Watson, and M. Westhead, "The design and implementation of grid database services in OGSA-DAI," *Concurrency and Computation: Practice and Experience*, vol. 17, pp. 357–376, Feb. 2005.

[42] "Open Middleware Infrastructure Institute UK (OMII-UK)."
http://www.omii.ac.uk/.

[43] "Storage Resource Broker (SRB)." http://www.sdsc.edu/srb/index.php/Main_Page.

[44] "UNICORE: Distributed computing and data resources." http://www.unicore.eu/.

[45] "glite, a lightweight middleware for grid computing." http://glite.web.cern.ch/glite/.

[46] V. Breton, A. E. Solomonides, and R. H. McClatchey, "A perspective on the healthgrid initiative," in *Proceedings of Cluster Computing and the Grid (CCGrid) 2004*, pp. 434–439, Apr. 2004.

[47] V. Breton, I. Blanquer, V. Hernandez, N. Jacq, Y. Legr, M. Olive, T. Solomonides, H. Rahmouni, I. Andoulsi, J. Herveg, and P. Wilson, "SHARE roadmap 1: Towards a debate," in *Proceedings of Healthgrid 2007 conference*, 2007.

[48] V. Hernandez and I. Blanquer, "The grid as a healthcare provision tool," *Methods of Information in Medicine*, vol. 44, pp. 144–148, 2005.

[49] C. P. Waegemann, "Electronic health record." Status report 2002, Medical Record Institute.

[50] T. Handler, R. Holtmerier, J. Metzger, M. Overhage, S. Taylor, and Underwood, "HIMSS electronic health record, definitional model version 1.0." 2003.

[51] "NHS choices - your health, your choices." http://www.nhs.uk.

[52] "Information for health: an information strategy for the modern NHS 1998-2005." National HEALTH Service, UK, 1998.

[53] "Building the information core: Implementing the nhs plan." 2001.

[54] "Health Level Seven (HL7)." http://www.hl7.org/.

[55] "Open Electronic Health Records (OpenEHR)." http://www.openehr.org/.

[56] "DELOS: A network of excellence in digital libraries." http://www.delos.info.

[57] R. Davis, H. Shrobe, and P. Szolovits, "What is a knowledge representation?," *AI Magazine*, vol. 14(1), p. 1733, 1993.

[58] C. Brewster and W. Yorick, "Ontologies, taxonomies, thesauri: Learning from texts," in *Proceedings of The Use of Computational Linguistics in the Extraction of Keyword Information from Digital Library Content Workshop*, (Kings College, London, UK), 2004.

[59] L. M. Garshol, "Metadata? thesauri? taxonomies? topic maps! making sense of it all," *Journal of Information Science*, vol. 30(4), pp. 378–391, 2004.

[60] S. Grimm, P. Hitzler, and A. Andreas, "Knowledge representation and ontologies," *Semantic Web Services: Concept, Technology and Application*, pp. 51–106, 2007.

[61] C. A. Welty, "An integrated representation for software development and discovery." PhD Thesis, Rensselaer Polytechnic Institute, 1995.

[62] "Extensible Markup Language (XML)." http://www.w3.org/XML/.

[63] "Simple HTML Ontology Extensions (SHOE)." http://www.cs.umd.edu/projects/plus/SHOE/.

[64] "Resource Description Framework (RDF)." http://www.w3.org/RDF/.

[65] "Web Ontology Language (OWL)." http://www.w3.org/2004/OWL/.

[66] "W3C: World Wide Web Consortium." http://www.w3.org/.

[67] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, "The description logic handbook: Theory, implementation, applications," *Cambridge University Press*, 2003. ISBN 0-521-78176-0.

[68] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American Magazine*, May 2001.

[69] D. D. Roure, N. R. Jennings, and N. R. Shadbolt, "The semantic grid: Past, present, and future," in *Proceedings of the IEEE*, vol. 93, p. 669681, Mar. 2005.

[70] C. A. Goble, D. D. Roure, N. R. Shadbolt, and A. A. A. Fernandes, *Enhancing Services and Applications with Knowledge and Semantics*. 2004.

[71] J. Brooke, K. Garwood, and C. Goble, "Semantic matching of grid resource descriptions," in *Proceedings of Second European Across Grids Conference (AXGrids 2004)*, (Nicosia, Cyprus), p. 240249, 2004. LNCS 3165.

[72] P. Alper, O. Corcho, I. Kotsiopoulos, P. Missier, S. Bechhofer, D. Kuo, and C. Goble, "S-OGSA as a reference architecture for OntoGrid and for the semantic grid." The 3rd GGF Semantic Grid Workshop, GGF16, 2006.

[73] L. Pouchard, L. Cinquini, and G. Strand, "The earth system grid discovery and semantic web technologies." Semantic Web Technologies for Searching and Retrieving Scientific Data (ISWCII), Oct. 2003.

[74] S. Miles, J. Papay, V. Dialani, M. Luck, K. Decker, T. Payne, and L. Moreau, "Personalised grid service discovery," in *IEE Proceedings Software: Special Issue on Performance Engineering*, vol. 150, 2003.

[75] M. Babik, E. Gatial, O. Habala, E. Hluchy, M. Laclavik, and M. Maliska, "Semantic grid services in k-wf grid." Second International Conference on Semantics, Knowledge and Grid, Nov. 2006.

[76] C. Zhu, Z. Liu, W. M. Zhang, W. D. Xiao, and J. C. Huang, "An efficient decentralized grid service discovery approach based on service ontology," in *Proceedings of IEEE/WIC/ACM International Conference*, vol. 20-24, Sept. 2004.

[77] W. Xing, M. D. Dikaiakos, and R. Sakellariou, "A core grid ontology for the semantic grid," in *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, pp. 178–184, 2006.

[78] "Dublin core metadata initiative." http://dublincore.org/.

[79] G. T. Henderson, E. C. Ifeachor, H. S. K. Wimalartna, A. E., and N. R. Hudson, "Prospects for routine detection of dementia using the fractal dimension of the human electroencephalogram," pp. 284–289, 2000.

[80] P. Zhao, P. Van Eetvelt, C. Goh, N. Hudson, S. Wimalaratna, and E. C. Ifeachor, "EEG markers of azheimers disease using tsallis entropy," in *3rd International Conference on Computational Intelligence in Medicine and Healthcare (CIMED2007)*, (Plymouth, U.K), July 2007.

[81] "The Globus alliance." http://www.globus.org.

[82] "BioinfoGRID, bioinformatics grid application for life science." http://www.bioinfogrid.eu/.

[83] B. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, "The Globus striped GridFTP framework and server." SC2005, 2005.

[84] Condor Project http://www.cs.wisc.edu/condor/.

[85] J. M. Schopf, I. Raicu, L. Pearlman, N. Miller, C. Kesselman, and I. Foster, "Monitoring and discovery in a web services framework: Functionality and performance of the Globus toolkits MDS4." ANL/MCS-P1248-0405.

[86] A. Nadalin, C. Kaler, P. H. Baker, and R. Monzillo, "Web service security: SOAP message security 1.0 (WS-Security 2004)." Mar. 2004.

[87] "Information technology open systems interconnection – the directory: Authentication framework." Series X: Data Networks and Open System Communications, ITU Recommendation X.509, Aug. 1997.

[88] E. C. Ifeachor, A. Anastasiou, C. Goh, N. Outram, M. Zervakis, S. V. Huffel, D. Timmerman, P. Lisboa, A. Taktak, A. Starita, K. Cerbiono, F. Ferreira, and G. Balls, "A data model to support bioprofiling." White Paper, BIOPATTERN Project, Jan. 2007.

[89] J. Broekstra, A. Kampman, and F. V. Harmelen, "Sesame, a generic architecture for storing and querying RDF and RDF schema," in *International Semantic Web Conference 2002*, (Sardinia, Italy).

[90] "Jena: A semantic web framework for java." http://jena.sourceforge.net/.

[91] P. Zhao, P. Van-Eevelt, C. Goh, N. Hudson, S. Wimalaratna, and E. C. Ifeachor, "Characterization of EEGs in Alzheimers disease using information theoretic methods," in *Proceedings of the 29th IEEE EMBS Annual International Conference*, (Cit Internationale, Lyon, France), Aug. 2007.

[92] P. Ziegler and K. R. Dittrich, "Three decades of data integration all problems solved?," *18th IFIP World Computer Congress (WCC2004)*, vol. 12, pp. 3–12, Aug. 2004.

[93] D. S. Knopman, S. T. DeKosky, J. L. Cummings, H. Chui, J. Corey-Bloom, N. Relkin, G. W. Small, B. Miller, and J. C. Stevens, "Practice parameter: diagnosis of dementia (an evidence-based review): report of the quality standards subcommittee of the american academy of neurology," vol. 56, no. 9, pp. 1143–1153, 2001.

[94] B. Hjorth, "Eeg analysis based on time domain properties," *Electroencephalography and Clinical Neurophysiology*, vol. 29, p. 306310, Sept. 1970.

[95] "Plymgrid, a campus grid at the university of plymouth." http://ils413.uopnet.plymouth.ac.uk/.

[96] "The BIOPATTERN grid portal." https://141.163.121.185:8080.

[97] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A security architecture for computational grids," in *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pp. 83–92, Nov. 1998.

[98] M. Aldinucci, M. Danelutto, A. Paternesi, R. Ravazzolo, and M. Vanneschi, "Building interoperable grid-aware assist applications via web services," in *Parallel Computing Conference*, Sept. 2005.

[99] M. Lettere, D. Guerri, and R. Fontanelli, "Prototypal ambient intelligence framework for assessment of food quality and safety," in *9th Int. Congress of the Italian Association for Artificial Intelligence (AI*IA 2005) Advances in artificial Intelligence*, (Milan, Italy), pp. 442–453, Sept. 2005.

[100] C. J. Tjhai, M. Tomlinson, R. Horan, M. Ahmed, and M. Ambroze, "On the efficient codewords counting algorithm and the weight distributions of the binary quadratic double circulant codes," in *Proceedings of IEEE Information Theory Workshop*, (Chengdu, China), Oct. 2006.

[101] "SuSE Linux operating systems." http://www.novell.com/linux/.

[102] T. Friese, S. G., and B. Freisleben, "GDT: A toolkit for grid service development," in *Proceeding of the 3rg International Conference on Grid Service Engineering and Management*, pp. 131–148, 2006.

[103] "JavaServer Pages (JSP) technology." http://java.sun.com/products/jsp/.

[104] "HyperText Markup Language (html)." http://www.w3.org/MarkUp/.

[105] "Apache Tomcat project." http://tomcat.apache.org/.

[106] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Service Description Language (WSDL) 1.1." W3C Note, http://www.w3.org/TR/wsdl, Mar. 2001.

[107] "OWL-S: Semantic markup for web services." The OWL Services Coalition, Technical Report, http://www.daml.org/services/owl-s/1.0/owl-s.html, Dec. 2003.

[108] "The Protege ontology editor and knowledge acquisition system." http://protege.stanford.edu/.

[109] "OWLViz: A visualisation plugin for the Protg OWL plugin." http://www.co-ode.org/downloads/owlviz/co-ode-index.php.

[110] "RacerPro: An OWL reasoner and inference server for the semantic web." http://www.racer-systems.com/.

[111] "A grid kernel ontology skeleton for service-oriented grids." https://141.163.121.185:8080/download/gridKernelOntoSkeleton.owl.

[112] R. Allan, X. D. Wang, A. Richards, and D. Chohan, "XML schema for-e-science projects, grid users, applications and resources." IeSE Developers' Notes (IeSE-3), Apr. 2007.

[113] I. Kojima, "Design and implementation of OGSADAI-RDF." GCF16 Semantic Grid Workshop.

Appendix A

Examples of Database Schema used in Description of Bioprofile Information

A.1 Patient Info Schema

The patient information schema contains 11 attributes of contents to describe a patient, including patient's ID: patient_id, first name: first_name, middle and last name: middle_last_name, date of birth: date_of_birth, gender: gender, ethnic origin: ethnic_origin, city and country of birth: city_origin and country_origin, city and country of current resident: city_origin and country_origin, and nationality: nationality.

```
CREATE TABLE patient
(
    patient_id          VARCHAR(250)    PRIMARY KEY,
    first_name          VARCHAR(250),
    middle_last_name    VARCHAR(250),
    date_of_birth       DATE,
    gender              CHAR(1)         CHECK(gender IN ('M', 'F')),
    ethnic_origin       VARCHAR(250),
    city_origin         VARCHAR(250),
    country_origin      VARCHAR(250),
    city_of_resident    VARCHAR(250),
    country_of_resident VARCHAR(250),
    nationality         VARCHAR(250),
);
```

A.2 EEG Data Schema

The EEG data schema contains 10 attributes of contents to describe an EEG data file, including EEG data file ID: dataset_id, which patient an EEG data file belongs to: belong_to, EEG data file size: dataset_size, recorded length of an EEG data file (sec.): eeg_length, number of channel used in recording of an EEG data file: number_of_channel, what format of an EEG data file is: dataset_format, the location of an EEG data file: location, date of the creation of an EEG data file: create_date, date of the modification of an EEG data file: modify_date, text description to an EEG data file: description.

```
CREATE TABLE eeg_datasets
(
    dataset_id          VARCHAR(250)    PRIMARY KEY,
    belong_to           VARCHAR(250),
    dataset_size        INTEGER,
    eeg_length          INTEGER,
    number_of_channel   INTEGER,
    dataset_format      VARCHAR(50),
    location            VARCHAR(250),
    create_date         DATE,
    modify_date         DATE
    description         VARCHAR(250),
);
```

A.3 EEG Analysis Algorithm Info Schema

The EEG analysis algorithm information schema contains 6 attributes of contents to describe an EEG analysis algorithm, including the ID of an algorithm: algorithm_id, the name of an algorithm: algorithm_name, analysis type (e.g. fractal dimension and Tsallis entropy) of an algorithm: algorithm_type, the version of an algorithm: algorithm_version, the reference(s) (e.g. research paper and thesis) of an algorithm: algorithm_reference, text description of an algorithm: description.

```
CREATE TABLE algorithms
(
    algorithm_id            VARCHAR(250)     PRIMARY KEY,
    algorithm_name          VARCHAR(250),
    algorithm_type          VARCHAR(250),
    algorithm_version       VARCHAR(250),
    algorithm_reference     VARCHAR(250),
    description             VARCHAR(250),
);
```

A.4 EEG Analysis Software Implementations Info Schema

The EEG analysis software implementations information schema contains 8 attributes of contents to describe a software implementation of an EEG analysis algorithm, including the ID of an implementation: implementation_id, the ID of the reference algorithm of an implementation: algorithm_id, the implementation type (e.g. C executable and java classes): implementation_type, the version of an implementation: implementation_version, the location of an implementation: location, date of creation of an implementation: create_date, date of modification of an implementation: modify_date, and text description of an implementation: description.

```
CREATE TABLE algorithm_implementations
(
    implementation_id         VARCHAR(250)    PRIMARY KEY,
    algorithm_id              VARCHAR(250),
    implementation_type       VARCHAR(250),
    implementation_version    VARCHAR(250),
    location                  VARCHAR(250),
    create_date               DATE,
    modify_date               DATE,
    description               VARCHAR(250),
);
```

Appendix B

An Example of an XML based Mapping Schema

This example of a mapping schema shows the attribute mapping information between a standard EEG data schema and a legacy EEG data schema used at a bioprofile centre. In the mapping schema, the element "domain_map" is considered as the head information of mapping. It contains two sub-elements "src_domain" and "target_domain", which describe the legacy EEG data schema as the source schema needed to be mapped into the standard EEG data schema as the target schema, and the URIs of them. The elements "field_map" are considered as the body information. Each element "field_map" contains two sub-elements "src_field" and "target_field", which describe the mapping between an attribute name of the source schema and an attributed name of the target schema.

```
<?xmlversion="1.0" encoding="UTF-8"? >
<bg:mapping xmlns="http://www.w3.org/2001/XMLSchema"
 xmlns:bg="http://www.tech.plymouth.ac.uk/spmc/biopatternGrid/">
      <bg:map>
         <bg:domain_map>
            <bg:src_domain>
                http://biopatternGrid.com/TSI/bioprofile/
                clinical/metadata#datasets
            </bg:src_domain>
            <bg:target_domain>
                http://biopatternGrid.com/standard/bioprofile/
                electrophysiology/EEG/metadata#eeg_datasets
            </bg:target_domain>
         </bg:domain_map>
         <bg:field_map>
            <bg:src_field>eeg_length</bg:src_field>
```

```
        <bg:target_field>eeg_duration</bg:target_field>
      </bg:field_map>
      <bg:field_map>
        <bg:src_field>number_of_channel</bg:src_field>
        <bg:target_field>total_channel</bg:target_field>
      </bg:field_map>
      <bg:field_map>
        <bg:src_field>dataset_size</bg:src_field>
        <bg:target_field>eeg_file_size</bg:target_field>
      </bg:field_map>
    </bg:map>
</bg:mapping>
```

Appendix C

An Example of an XML based Mapping Schema Registry

This example of a mapping schema registry is an XML document, which can be published, for example, via the GT4 default index services. The registry shows "domain_map" information, which is the same as those contained in each mapping schema in order to let users query for the specific mapping schema they want from index services. The registry also uses the element "location", which comes with each "domain_map" element, to present the location of each mapping schema to users. Therefore, users will be able to retrieve required mapping schemas based on the obtained location information.

```
<?xmlversion="1.0" encoding="UTF-8"? >
<bg:mapping_registry xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:bg="http://www.tech.plymouth.ac.uk/spmc/biopatternGrid/">
      <bg:map>
         <bg:domain_map>
            <bg:src_domain>
                http://biopatternGrid.com/TSI/bioprofile/
                clinical/metadata#datasets
            </bg:src_domain>
            <bg:target_domain>
                http://biopatternGrid.com/standard/bioprofile/
                electrophysiology/EEG/metadata#eeg_datasets
            </bg:target_domain>
         </bg:domain_map>
         <bg:location>
            gsiftp://141.163.121.40:2811/home/globus/mappings/
            eeg_tsi2bgstanrdard.xml
         </bg:location>
```

```
    </bg:map>
    <bg:map>
        <bg:domain_map>
            <bg:src_domain>
                http://biopatternGrid.com/TUT/bioprofile
                /medical/metadata#datasets
            </bg:src_domain>
            <bg:target_domain>
                http://biopatternGrid.com/standard/bioprofile/
                electrophysiology/EEG/metadata#eeg_datasets
            </bg:target_domain>
        </bg:domain_map>
        <bg:location>
            gsiftp://141.163.121.40:2811/home/globus/mappings/
            eeg_tut2bgstanrdard.xml
        </bg:location>
    </bg:map>
<bg:mapping_registry>
```

Appendix D

An Example of an OWL based Description of an Application

This example of a description of an application is implemented based on the K-WF grid project. The example contains certain major classes to represent main entities of an application, including "Description", "Process" and "AppID". The "Description" class contains two sub-classes "Name" and "Version" to describe application name and version information. The Process class contains four sub-classes "Input", "Output", "Requirements" and "Function" to describe information of application input (e.g. ID of a specific analysis algorithm), output (e.g. a type of analysis result), requirements, and functions (e.g. Hjorth algorithm based EEG analysis and subject information query), respectively.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns# "
xmlns:xsd="http://www.w3.org/2001/XMLSchema# "
xmlns:owltime="http://www.isi.edu/ pan/damltime/time-entry.owl# "
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema# "
xmlns:owl="http://www.w3.org/2002/07/owl# "
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns="http://gom.kwfgrid.net/ontology/ApplicationOntology# "
xml:base="http://gom.kwfgrid.net/ontology/ApplicationOntology">
<owl:Ontology rdf:about="">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Application generic K-Wf Grid ontology. </rdfs:comment>
<owl:imports rdf:resource="http://gom.kwfgrid.net/ontology/
public/time/time-entry.owl# "/>
</owl:Ontology>
<owl:Class rdf:ID="Cost">
```

```
<rdfs:subClassOf>
<owl:Class rdf:ID="ComparisonCriteria"/>
</rdfs:subClassOf>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
The concept of abstract Cost (it can be Time, Money, etc.)</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Transformation">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string"
>
An activity that performs transformation.</rdfs:comment>
<rdfs:subClassOf>
<owl:Class rdf:ID="Activity"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Description"/>
<owl:Class rdf:ID="Prediction">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
An activity that performs prediction.</rdfs:comment>
<rdfs:subClassOf>
<owl:Class rdf:about="# Activity"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="SimulationParameter">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
The parameter of simulation activity</rdfs:comment>
<rdfs:subClassOf>
```

```
<owl:Class rdf:ID="ActivityParameter"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="URL"/>
<owl:Class rdf:ID="Reliability">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
</rdfs:comment>
<rdfs:subClassOf>
<owl:Class rdf:about="# ComparisonCriteria"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="AppID"/>
<owl:Class rdf:ID="Simulation">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
An activity that performs simulation.</rdfs:comment>
<rdfs:subClassOf>
<owl:Class rdf:about="# Activity"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Version">
<rdfs:subClassOf rdf:resource="# Description"/>
</owl:Class>
<owl:Class rdf:about="# ActivityParameter">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
An abstract concept representing parameters of the activity
. The subclasses or individuals of this concept could be later reused
```

```
as parameters of services invocations.</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Requirements">
<rdfs:subClassOf>
<owl:Class rdf:ID="Process"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Output">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="isResultOf"/>
</owl:onProperty>
<owl:allValuesFrom>
<owl:Class rdf:ID="Input"/>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="# Process"/>
</owl:Class>
<owl:Class rdf:ID="UserRating">
<rdfs:subClassOf>
<owl:Class rdf:about="# ComparisonCriteria"/>
</rdfs:subClassOf>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
The average rating of users of some entity
```

```
(e.g. a service, workflow...)</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="# Activity">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
An abstract concept for representing activities.</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Function">
<rdfs:subClassOf rdf:resource="# Process"/>
</owl:Class>
<owl:Class rdf:ID="Actuation">
<rdfs:subClassOf rdf:resource="# Activity"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
An activity that performs actuation of some value,
e.g. an update of control.</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Speed">
<rdfs:subClassOf>
<owl:Class rdf:about="# ComparisonCriteria"/>
</rdfs:subClassOf>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Monitoring">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
An activity that performs monitoring.</rdfs:comment>
<rdfs:subClassOf rdf:resource="# Activity"/>
```

```
</owl:Class>
<owl:Class rdf:about="# ComparisonCriteria">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
An abstract concept of an Comparison Criterium. These are
usually set by users and possibly adress both
services as well as workflows</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="# Input">
<rdfs:subClassOf rdf:resource="# Process"/>
</owl:Class>
<owl:Class rdf:ID="Name">
<rdfs:subClassOf rdf:resource="# Description"/>
</owl:Class>
<owl:Class rdf:ID="Analysis">
<rdfs:subClassOf rdf:resource="# Activity"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
An activity that performs analysis.</rdfs:comment>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasDivision">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
A ation of aggregation between and Organization and Division</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isPartOf">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
A ation between a UserGroup and Virtual Organization to
which it belongs</rdfs:comment>
```

```
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasAuthor">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Author of the text note.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="involvesPerson">
<rdfs:domain rdf:resource="# Activity"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
An abstract property for ating users involved in activities.<
/rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasParameter">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
An abstract property for ating parameters with activities<
/rdfs:comment>
<rdfs:range rdf:resource="# ActivityParameter"/>
<rdfs:domain rdf:resource="# Activity"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isMemberOf">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Relates people to various groups</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasComparisonCriterium">
<rdfs:domain rdf:resource="# Activity"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
The criterium associated with the activity, e.g. a Cost or UserRating<
```

```
/rdfs:comment>
<rdfs:range rdf:resource="# ComparisonCriteria"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="inRegion">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
ates the location with a region.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="involvesOrganization">
<rdfs:domain rdf:resource="# Activity"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
An abstract property for ating organizations involved in
activities.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="participatesIn">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
A ation of aggregation between VirtualOrganization and Organization<
/rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="administeredBy">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
The administrator/person responsible for the Virtual Organization<
/rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasDescription"/>
<owl:DatatypeProperty rdf:ID="url">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema# anyURI"/>
```

```
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Person's website with possibly additional information.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNationality">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema# string"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Person's nationality.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="name">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema# string"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
The name of the Virtual Organization</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasEmail">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema# string"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Person's preffered contact e-mail.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasPhoneNumber">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Person's preferred contact phone.</rdfs:comment>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema# string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasFirstName">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Person's first name.</rdfs:comment>
```

```
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema# string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="text">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
The free text contained in the Note</rdfs:comment>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema# string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="description">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema# string"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Short textual description of the Virtual Organization,
e.g. purpose, domain, etc.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="unitRating">
<rdfs:domain rdf:resource="# UserRating"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema# float"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Rating expressed as a float value between 0.0 and 1.0</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasLastName">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema# string"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Person's last name.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasTitle">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
```

```
Persons's title - e.g. PhD</rdfs:comment>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema# string"/>
</owl:DatatypeProperty>
<Input rdf:ID="SubjectID">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
ID of a subject (e.g. patient)</rdfs:comment>
</Input>
<AppID rdf:ID="EA4EDD_01435"/>
<owl:AnnotationProperty rdf:ID="htmlForm"/>
<Input rdf:ID="SpecificAnalysisAlgorithmID">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
The ID(s) of user-required analysis algorithm(s)/method(s)<
/rdfs:comment>
</Input>
<Function rdf:ID="SubInfoAccess">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Access to subject information</rdfs:comment>
</Function>
<Function rdf:ID="EEG_FD_Analysis">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
EEG analysis based on fractal dimension algorithm</rdfs:comment>
</Function>
<Function rdf:ID="EEG_Hjorth_Analysis">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
EEG analysis based on Hjorth algorithm</rdfs:comment>
</Function>
```

```
<Output rdf:ID="SubDescription">
<isResultOf rdf:resource="# SubjectID"/>
</Output>
<Version rdf:ID="EA4EDD_alfa">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Alfa version of EEG analysis for early detection of dementia<
/rdfs:comment>
</Version>
<Function rdf:ID="EEG_TE_Analysis">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
EEG Analysis based on Tsallis Entropy</rdfs:comment>
</Function>
<owl:AnnotationProperty rdf:about="http://purl.org/dc/elements/1.1/identifier"/>
<Output rdf:ID="AnalysisResults">
<isResultOf rdf:resource="# SpecificAnalysisAlgorithmID"/>
</Output>
<Name rdf:ID="EEGAnalysisForEarlyDetectionOfDementia">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
EEG analysis for early detection of dementia</rdfs:comment>
</Name>
</rdf:RDF>
```

Appendix E

Examples of XML based Descriptions of Partners

E.1 An Example of an XML based Description of a Grid Partner

This example of a description of a grid partner is implemented based on the IeSE person schema. The example describes a grid partner (in this case, a grid user account) in two aspects, as personal information and as the partner's role in a grid environment. The personal information is described by the element "personName" and "personDesc". The element "personDesc" further uses sub-elements "personKey", "shortDescription", "contactEMail" and "contactAddress" to describe the partner's identification/reference, text description, email and address used for contact, respectively. The partner's role is described by the element "roleKey", "voName" and "roleDescription" to indicate the identification/reference of the role that the grid partner plays, the text description to the role, and which VO the grid partner belongs to, respectively.

```
<?xmlversion="1.0" encoding="UTF-8"? >
<person xmlns="http://www.w3.org/2001/XMLSchema">
      <personName>
         EEG analysis clinician group user
      </personName>
         <personDesc>
         <personKey>clinician_0137532</personKey>
         <shortDescription>
            permitted clinicians for the use of distributed EEG analysis
         </shortDescription>
```

```
        <contactEMail>
           eeganalysisapplicationadmin@biopattern.org
        </contactEMail>
        <contactAddress>Biopattern
           Network of Excellence, University of Plymouth,
           Plymouth, UK
        </contactAddress>
    </personDesc>
    <personrole>
        <roleKey>clinicianEEGAnalysisBGVO1</roleKey>
        <voName>BIOPATTERNGridVO1</voName>
        <roleDescription>
           a shared user account for clinicians to carry out
           EEG analysis over grid
        </roleDescription>
    </personrole>
</person>
```

E.2 An Example of XML based Description of an Application Partner

This example of a description of an application partner is implemented partly based on the draft of bioprofiling data model. It includes three main elements: "personID", "personDes" and "personRole", which describes the ID of an application partner, the application partner, and his/her role(s) in application(s), respectively. The element "personDes" composes of certain elements, such as "surname", "firstname", "title", "employer", "position", "staffed", "contactAddress", and "contactEmail", which describe the name and title of the application partner, his/her employer, occupational position, employer ID and contact details, respectively. The element "personRole" consists of two elements, "roleID" and "roleDes" to indicate the reference of the role in an application the partner plays, and the text description of his/her role in the application, respectively.

```
<?xmlversion="1.0" encoding="UTF-8"? >
<bg:person xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:bg="http://www.tech.plymouth.ac.uk/spmc/biopatternGrid/">
    <bg:personID>bg_uop_007</bg:personID>
    <bg:personDes>
        <bg:surname>Bond</bg:surname>
        <bg:firstname>James</bg:firstname>
        <bg:title>Dr</bg:title>
        <bg:employer>
            <bg:employerName>Derriford Hospital</bg:employerName>
            <bg:employerAddress>
                Plymouth Hospitals NHS Trust,
                Derriford Road,
                Crownhill,
                Plymouth,
                Devon,
                UK,
                PL6 8DH
            </bg:employerAddress>
        </bg:employer>
        <bg:position>
            Resident physician
        </bg:position>
        <staffID>13579</staffID>
        <bg:contactAddress>
            PO Box 1300, London, UK, SE1 1BD
        </bg:contactAddress>
        <bg:contactEmail>james.bond@mi6.gov.uk</bg:contactEmail>
    </bg:personDes>
    <bg:personRole>
        <bg:roleID>clinician_uk_sw_group5c</bg:roleID>
        <bg:roleDes>
            A member of the clinician group 5c in south west of the UK.
        </bg:roleDes>
    </bg:personRole>
</bg:person>
```

Appendix F

An Example of a WSDL based Description of Service

This example of service description is implemented based on WSDL. It describes certain Client API accessible operations of a parallel job service and related message exchange interfaces. The operations include "jobSubAndRun" for submission and running of a parallel job, "jobStatus" for the query of status of a submitted job, "desctoryJob" for job destroy, and "givenServiceRegistryLocation" for the indication of client-preferred index services for the discovery of computational resources.

```
<?xml version="1.0" encoding="UTF-8"? >
<wsdl:definitions name="GridJobManagementService"
   targetNamespace="http://tech.plym.ac.uk/phu/grid/services/
   jobManagement/GridJobManagement"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:apachesoap="http://xml.apache.org/xml-soap"
   xmlns:impl="http://tech.plym.ac.uk/phu/grid/services/
   jobManagement/GridJobManagement"
   xmlns:intf="http://tech.plym.ac.uk/phu/grid/services/
   jobManagement/GridJobManagement"
   xmlns:tns="http://tech.plym.ac.uk/phu/grid/services/
   jobManagement/GridJobManagement"
   xmlns:tns1="http://beans.service.grid.phu"
   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdlpp="http://www.globus.org/namespaces/2004/10/
   WSDLPreprocessor"
   xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:wsntw="http://docs.oasis-open.org/wsn/2004/06/
```

```
wsn-WS-BaseNotification-1.2-draft-01.wsdl"
xmlns:wsrp="http://docs.oasis-open.org/wsrf/2004/06/
wsrf-WS-ResourceProperties-1.2-draft-01.xsd"
xmlns:wsrpw="http://docs.oasis-open.org/wsrf/2004/06/
wsrf-WS-ResourceProperties-1.2-draft-01.wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<wsdl:import location="../../wsrf/properties/WS-ResourceProperties.wsdl"
namespace="http://docs.oasis-open.org/wsrf/2004/06/
wsrf-WS-ResourceProperties-1.2-draft-01.wsdl"/>
<wsdl:import location="../../wsrf/notification/WS-BaseN.wsdl"
namespace="http://docs.oasis-open.org/wsn/2004/06/
wsn-WS-BaseNotification-1.2-draft-01.wsdl"/>
<wsdl:types>
  <schema elementFormDefault="qualified"
  targetNamespace="http://tech.plym.ac.uk/phu/grid/services/
  jobManagement/GridJobManagement"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:rpns="http://docs.oasis-open.org/wsn/2004/06/
  wsn-WS-BaseNotification-1.2-draft-01.xsd"
  xmlns:tns="http://tech.plym.ac.uk/phu/grid/services/
  jobManagement/GridJobManagement"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:import namespace="http://schemas.xmlsoap.org/
    ws/2004/03/addressing"
    schemaLocation="../../ws/addressing/WS-Addressing.xsd"/>
      <import namespace="http://beans.service.grid.phu"/>
```

```
<element name="jobSubAndRun">
 <complexType>
  <sequence>
    <element maxOccurs="unbounded"
    name="stageInFileLocation"
    type="impl:ArrayOf_tns1_ArrayBean"/>
    <element maxOccurs="unbounded"
    name="stageOutFileLocation"
    type="impl:ArrayOf_tns1_ArrayBean"/>
    <element maxOccurs="unbounded"
    name="commandLine" type="impl:ArrayOf_tns1_ArrayBean"/>
    <element name="userName" type="xsd:string"/>
    <element name="userGroup" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<complexType name="ArrayOf_tns1_ArrayBean">
 <sequence>
  <element maxOccurs="unbounded" minOccurs="0"
  name="item" type="tns1:ArrayBean"/>
 </sequence>
</complexType>
<element name="jobSubAndRunResponse">
 <complexType>
  <sequence>
    <element name="jobSubAndRunReturn" type="xsd:string"/>
    </sequence>
  </complexType>
</element>
<element name="jobStatus">
  <complexType>
    <sequence>
       <element name="jobEprFileName" type="xsd:string"/>
    </sequence>
  </complexType>
</element>
<element name="jobStatusResponse">
  <complexType>
    <sequence>
       <element name="jobStatusReturn" type="xsd:string"/>
    </sequence>
  </complexType>
</element>
<element name="destroyJob">
  <complexType>
    <sequence>
       <element name="jobEprFileName" type="xsd:string"/>
    </sequence>
  </complexType>
</element>
<element name="destroyJobResponse">
  <complexType>
```

```
      <sequence>
         <element name="destroyJobReturn" type="xsd:string"/>
      </sequence>
    </complexType>
  </element>
  <element name="givenServiceRegistryLocation">
    <complexType>
      <sequence>
         <element maxOccurs="unbounded" name="serviceRegistryLocation"
         type="impl:ArrayOf_tns1_ArrayBean"/>
      </sequence>
    </complexType>
  </element>
  <element name="givenServiceRegistryLocationResponse">
    <complexType>
      <sequence>
         <element name="givenServiceRegistryLocationReturn"
         type="xsd:string"/>
      </sequence>
    </complexType>
  </element>
  <xsd:element name="totalJobSubmitted" type="xsd:int"/>
  <element name="GridJobManagementResourceProperties">
    <complexType>
      <sequence>
         <element maxOccurs="1" minOccurs="1"
```

```
            ref="tns:totalJobSubmitted"/>
          </sequence>
        </complexType>
      </element>
    </schema>
    <schema elementFormDefault="qualified"
    targetNamespace="http://beans.service.grid.phu"
    xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://tech.plym.ac.uk/phu/grid/
      services/jobManagement/GridJobManagement"/>
      <complexType name="ArrayBean">
        <sequence>
          <element name="doubleContext" type="xsd:double"/>
          <element name="floatContext" type="xsd:float"/>
          <element name="intContext" type="xsd:int"/>
          <element name="stringContext" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
    </schema>
  </wsdl:types>
  <wsdl:message name="jobStatusResponse">
    <wsdl:part element="impl:jobStatusResponse" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="givenServiceRegistryLocationResponse">
    <wsdl:part element="impl:givenServiceRegistryLocationResponse"
    name="parameters"/>
```

```
</wsdl:message>
<wsdl:message name="destroyJobResponse">
   <wsdl:part element="impl:destroyJobResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="jobStatusRequest">
   <wsdl:part element="impl:jobStatus" name="parameters"/>
</wsdl:message>
<wsdl:message name="destroyJobRequest">
   <wsdl:part element="impl:destroyJob" name="parameters"/>
</wsdl:message>
<wsdl:message name="givenServiceRegistryLocationRequest">
   <wsdl:part element="impl:givenServiceRegistryLocation" name="parameters"/>
</wsdl:message>
<wsdl:message name="jobSubAndRunRequest">
   <wsdl:part element="impl:jobSubAndRun" name="parameters"/>
</wsdl:message>
<wsdl:message name="jobSubAndRunResponse">
   <wsdl:part element="impl:jobSubAndRunResponse" name="parameters"/>
</wsdl:message>
<wsdl:portType name="GridJobManagementPortType"
wsdlpp:extends="wsrpw:GetResourceProperty wsntw:NotificationProducer
wsntw:NotificationProducer"
wsrp:ResourceProperties="tns:GridJobManagementResourceProperties">
   <wsdl:operation name="jobSubAndRun">
    <wsdl:input message="impl:jobSubAndRunRequest"
    name="jobSubAndRunRequest"/>
    <wsdl:output message="impl:jobSubAndRunResponse"
    name="jobSubAndRunResponse"/>
    </wsdl:operation>
    <wsdl:operation name="jobStatus">
      <wsdl:input message="impl:jobStatusRequest"
      name="jobStatusRequest"/>
      <wsdl:output message="impl:jobStatusResponse"
      name="jobStatusResponse"/>
    </wsdl:operation>
    <wsdl:operation name="destroyJob">
      <wsdl:input message="impl:destroyJobRequest"
      name="destroyJobRequest"/>
      <wsdl:output message="impl:destroyJobResponse"
      name="destroyJobResponse"/>
    </wsdl:operation>
    <wsdl:operation name="givenServiceRegistryLocation">
      <wsdl:input message="impl:givenServiceRegistryLocationRequest"
      name="givenServiceRegistryLocationRequest"/>
      <wsdl:output message="impl:givenServiceRegistryLocationResponse"
      name="givenServiceRegistryLocationResponse"/>
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

Appendix G

An Example of an OWL based Grid Kernel Ontology

This example of a grid kernel ontology is implemented based on the proposed grid kernel ontology skeleton. It registers all implemented descriptions of those grid entities defined by the grid kernel ontology skeleton. It also directly uses those properties and constraints defined by the ontology skeleton to describe necessary relationships between those grid entities. The main functions delivered by this example are described in Section 6.4 and 6.5.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns="http://www.biopattern.org/sp3_2/grid/ontology/kernelontoskeleton.owl# "
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns# "
xmlns:xsd="http://www.w3.org/2001/XMLSchema# "
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema# "
xmlns:owl="http://www.w3.org/2002/07/owl# "
xml:base="http://www.biopattern.org/sp3_2/grid/ontology/kernelontoskeleton.owl">
<owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="ApplicationPartner">
<owl:disjointWith>
<owl:Class rdf:ID="GridPartner"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Class rdf:ID="Partner"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="QoSPolicy">
<rdfs:subClassOf>
<owl:Class rdf:ID="QoS"/>
```

```
</rdfs:subClassOf>
<owl:disjointWith>
<owl:Class rdf:ID="QoSEnforcement"/>
</owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="ApplicationDescription">
<owl:disjointWith>
<owl:Class rdf:ID="ApplicationDescriptionSchema"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="NamedApplication"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Restriction>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="useDescriptionSchema"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
<owl:allValuesFrom>
<owl:Class rdf:about="# ApplicationDescriptionSchema"/>
```

```
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:ID="Application"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="VODescription">
<owl:disjointWith>
<owl:Class rdf:ID="NamedVO"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="VODescriptionSchema"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Restriction>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
<owl:allValuesFrom>
<owl:Class rdf:about="# VODescriptionSchema"/>
```

```
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:ID="VO"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="# QoSEnforcement">
<rdfs:subClassOf>
<owl:Class rdf:about="# QoS"/>
</rdfs:subClassOf>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Implemented mechanisms, which realise individual or
a set of QoS policies.</rdfs:comment>
<owl:disjointWith rdf:resource="# QoSPolicy"/>
</owl:Class>
<owl:Class rdf:ID="GridJob">
<owl:disjointWith>
<owl:Class rdf:ID="GridMiddleware"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="VirtualResource"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Class rdf:ID="kernelOntoSkeleton"/>
</rdfs:subClassOf>
```

```
<owl:disjointWith>
<owl:Class rdf:ID="Security"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Partner"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="GridConnection"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# VO"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="GridNode"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="Service"/>
</owl:disjointWith>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
The abstraction of a paticular unit of work, which can solve
a/multiple problem(s) defined by a grid application.</rdfs:comment>
<owl:disjointWith>
<owl:Class rdf:about="# QoS"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Application"/>
```

```
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="PhysicalResource"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="Support"/>
</owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="ContainerDescriptionSchema">
<rdfs:subClassOf>
<owl:Class rdf:ID="Container"/>
</rdfs:subClassOf>
<owl:disjointWith>
<owl:Class rdf:ID="ContainerDescription"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="ContainerInstance"/>
</owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="GMDescriptionSchema">
<owl:disjointWith>
<owl:Class rdf:ID="GMDescription"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="NamedGridMiddleware"/>
</owl:disjointWith>
```

```
<rdfs:subClassOf>
<owl:Class rdf:about="# GridMiddleware"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="QEDescriptionSchema">
<owl:disjointWith>
<owl:Class rdf:ID="NamedQoSEnforcement"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="QEDescription"/>
</owl:disjointWith>
<rdfs:subClassOf rdf:resource="# QoSEnforcement"/>
</owl:Class>
<owl:Class rdf:about="# Support">
<rdfs:subClassOf rdf:resource="# kernelOntoSkeleton"/>
<owl:disjointWith rdf:resource="# GridJob"/>
<owl:disjointWith>
<owl:Class rdf:about="# QoS"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Partner"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# GridConnection"/>
</owl:disjointWith>
<owl:disjointWith>
```

```
<owl:Class rdf:about="# VO"/>
</owl:disjointWith>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Additional entities which provide global support for
descriptions of grid components, such as user-defined datatypes.</rdfs:comment>
<owl:disjointWith>
<owl:Class rdf:about="# GridNode"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# VirtualResource"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Security"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Service"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Application"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# PhysicalResource"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# GridMiddleware"/>
</owl:disjointWith>
```

```
</owl:Class>
<owl:Class rdf:ID="GCDescriptionSchema">
<rdfs:subClassOf>
<owl:Class rdf:about="# GridConnection"/>
</rdfs:subClassOf>
<owl:disjointWith>
<owl:Class rdf:ID="GCDescription"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="GridConnectionInstance"/>
</owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="# ContainerDescription">
<owl:disjointWith rdf:resource="# ContainerDescriptionSchema"/>
<owl:disjointWith>
<owl:Class rdf:about="# ContainerInstance"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Restriction>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
```

```
<owl:allValuesFrom rdf:resource="# ContainerDescriptionSchema"/>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="# Container"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="NamedSecurityPolicy">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="hasDescription"/>
</owl:onProperty>
<owl:allValuesFrom>
<owl:Class rdf:ID="SPDescription"/>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:ID="SecurityPolicy"/>
</rdfs:subClassOf>
<owl:disjointWith>
<owl:Class rdf:about="# SPDescription"/>
</owl:disjointWith>
<owl:disjointWith>
```

```
<owl:Class rdf:ID="SPDescriptionSchema"/>
</owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="NamedPhysicalResource">
<owl:disjointWith>
<owl:Class rdf:ID="PRDescription"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="PRDescriptionSchema"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="hasQoSEnforcement"/>
</owl:onProperty>
<owl:allValuesFrom>
<owl:Class rdf:about="# NamedQoSEnforcement"/>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:ID="NamedQoSPolicy"/>
</owl:allValuesFrom>
<owl:onProperty>
```

```
<owl:ObjectProperty rdf:ID="hasQoSPolicy"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="hasSecurityEnforcement"/>
</owl:onProperty>
<owl:allValuesFrom>
<owl:Class rdf:ID="NamedSecurityEnforcement"/>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# NamedSecurityPolicy"/>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="hasSecuritypolicy"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:about="# PRDescription"/>
```

```
</owl:allValuesFrom>
<owl:onProperty rdf:resource="# hasDescription"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="# PhysicalResource"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="VRDescription">
<owl:disjointWith>
<owl:Class rdf:ID="NamedVirtualResource"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="VRDescriptionSchema"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:about="# VRDescriptionSchema"/>
```

```
</owl:allValuesFrom>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="# VirtualResource"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="NamedService">
<owl:disjointWith>
<owl:Class rdf:ID="ServiceDescription"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="ServiceDescriptionSchema"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# NamedService"/>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="requireService"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasQoSEnforcement"/>
```

```
<owl:allValuesFrom>
<owl:Class rdf:about="# NamedQoSEnforcement"/>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasSecurityEnforcement"/>
<owl:allValuesFrom>
<owl:Class rdf:about="# NamedSecurityEnforcement"/>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:about="# ServiceDescription"/>
</owl:allValuesFrom>
<owl:onProperty rdf:resource="# hasDescription"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="# Service"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="# VRDescriptionSchema">
```

```
<rdfs:subClassOf>
<owl:Class rdf:about="# VirtualResource"/>
</rdfs:subClassOf>
<owl:disjointWith>
<owl:Class rdf:about="# NamedVirtualResource"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# VRDescription"/>
</owl:Class>
<owl:Class rdf:ID="APDescription">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
<owl:allValuesFrom>
<owl:Class rdf:ID="APDescriptionSchema"/>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="# ApplicationPartner"/>
<owl:disjointWith>
```

```
<owl:Class rdf:about="# APDescriptionSchema"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="NamedApplicationPartner"/>
</owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="# NamedQoSPolicy">
<owl:disjointWith>
<owl:Class rdf:ID="QPDescription"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="QPDescriptionSchema"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasDescription"/>
<owl:allValuesFrom>
<owl:Class rdf:about="# QPDescription"/>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="# QoSPolicy"/>
</owl:Class>
<owl:Class rdf:ID="SEDescriptionSchema">
<rdfs:subClassOf>
<owl:Class rdf:ID="SecurityEnforcement"/>
```

```
</rdfs:subClassOf>
<owl:disjointWith>
<owl:Class rdf:about="# NamedSecurityEnforcement"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="SEDescription"/>
</owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="# QPDescriptionSchema">
<owl:disjointWith rdf:resource="# NamedQoSPolicy"/>
<owl:disjointWith>
<owl:Class rdf:about="# QPDescription"/>
</owl:disjointWith>
<rdfs:subClassOf rdf:resource="# QoSPolicy"/>
</owl:Class>
<owl:Class rdf:about="# GMDescription">
<owl:disjointWith rdf:resource="# GMDescriptionSchema"/>
<owl:disjointWith>
<owl:Class rdf:about="# NamedGridMiddleware"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
</owl:Restriction>
```

```
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# GMDescriptionSchema"/>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="# GridMiddleware"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="# QEDescription">
<rdfs:subClassOf>
<owl:Restriction>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
<owl:allValuesFrom rdf:resource="# QEDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="# QoSEnforcement"/>
```

```
<owl:disjointWith>
<owl:Class rdf:about="# NamedQoSEnforcement"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# QEDescriptionSchema"/>
</owl:Class>
<owl:Class rdf:about="# SPDescription">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasDescription"/>
<owl:allValuesFrom>
<owl:Class rdf:about="# SPDescriptionSchema"/>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="# SecurityPolicy"/>
</rdfs:subClassOf>
<owl:disjointWith rdf:resource="# NamedSecurityPolicy"/>
<owl:disjointWith>
```

```
<owl:Class rdf:about="# SPDescriptionSchema"/>
</owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="# ServiceDescription">
<rdfs:subClassOf>
<owl:Restriction>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:about="# ServiceDescriptionSchema"/>
</owl:allValuesFrom>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="# Service"/>
</rdfs:subClassOf>
<owl:disjointWith rdf:resource="# NamedService"/>
<owl:disjointWith>
<owl:Class rdf:about="# ServiceDescriptionSchema"/>
</owl:disjointWith>
```

```
</owl:Class>
<owl:Class rdf:about="# QPDescription">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# QPDescriptionSchema"/>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="# QoSPolicy"/>
<owl:disjointWith rdf:resource="# NamedQoSPolicy"/>
<owl:disjointWith rdf:resource="# QPDescriptionSchema"/>
</owl:Class>
<owl:Class rdf:about="# Container">
<owl:disjointWith>
<owl:Class rdf:ID="Client"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Class rdf:about="# GridNode"/>
</rdfs:subClassOf>
```

```
</owl:Class>
<owl:Class rdf:about="# Application">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
A collection of work items that can carry out complex computing
tasks by using grid services and resources.</rdfs:comment>
<owl:disjointWith>
<owl:Class rdf:about="# QoS"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# GridConnection"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# VirtualResource"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# GridNode"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Service"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# PhysicalResource"/>
</owl:disjointWith>
<rdfs:subClassOf rdf:resource="# kernelOntoSkeleton"/>
<owl:disjointWith rdf:resource="# GridJob"/>
<owl:disjointWith>
```

```
<owl:Class rdf:about="# Partner"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Security"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# Support"/>
<owl:disjointWith>
<owl:Class rdf:about="# GridMiddleware"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# VO"/>
</owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="# NamedApplicationPartner">
<owl:disjointWith rdf:resource="# APDescription"/>
<owl:disjointWith>
<owl:Class rdf:about="# APDescriptionSchema"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="isA"/>
</owl:onProperty>
<owl:allValuesFrom>
<owl:Class rdf:ID="NamedGridPartner"/>
</owl:allValuesFrom>
```

```
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# APDescription"/>
<owl:onProperty rdf:resource="# hasDescription"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="# ApplicationPartner"/>
</owl:Class>
<owl:Class rdf:about="# NamedApplication">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasSecuritypolicy"/>
<owl:allValuesFrom rdf:resource="# NamedSecurityPolicy"/>
</owl:Restriction>
</rdfs:subClassOf>
<owl:disjointWith rdf:resource="# ApplicationDescription"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="requireVirtualResource"/>
</owl:onProperty>
<owl:allValuesFrom>
<owl:Class rdf:about="# NamedVirtualResource"/>
</owl:allValuesFrom>
```

```
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# NamedApplicationPartner"/>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="hasPartner"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# requireService"/>
<owl:allValuesFrom rdf:resource="# NamedService"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasDescription"/>
<owl:allValuesFrom rdf:resource="# ApplicationDescription"/>
</owl:Restriction>
</rdfs:subClassOf>
<owl:disjointWith>
<owl:Class rdf:about="# ApplicationDescriptionSchema"/>
</owl:disjointWith>
<rdfs:subClassOf>
```

```
<owl:Restriction>
<owl:onProperty rdf:resource="# hasQoSPolicy"/>
<owl:allValuesFrom rdf:resource="# NamedQoSPolicy"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:ID="NamedJobInstance"/>
</owl:allValuesFrom>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="archievedByJob"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="# Application"/>
</owl:Class>
<owl:Class rdf:about="# PhysicalResource">
<owl:disjointWith rdf:resource="# Support"/>
<owl:disjointWith>
<owl:Class rdf:about="# GridConnection"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# GridJob"/>
<owl:disjointWith>
<owl:Class rdf:about="# VirtualResource"/>
</owl:disjointWith>
```

```
<owl:disjointWith>
<owl:Class rdf:about="# GridNode"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# Application"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Any computing-related resources, such as a PC, a HPC,
a binary file, a data storage and communication networks.</rdfs:comment>
<owl:disjointWith>
<owl:Class rdf:about="# Service"/>
</owl:disjointWith>
<rdfs:subClassOf rdf:resource="# kernelOntoSkeleton"/>
<owl:disjointWith>
<owl:Class rdf:about="# GridMiddleware"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# VO"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Partner"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# QoS"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Security"/>
</owl:disjointWith>
```

```
</owl:Class>
<owl:Class rdf:ID="ClientDescription">
<rdfs:subClassOf>
<owl:Restriction>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:ID="ClientDescriptionSchema"/>
</owl:allValuesFrom>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="# Client"/>
</rdfs:subClassOf>
<owl:disjointWith>
<owl:Class rdf:about="# ClientDescriptionSchema"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="ClientInstance"/>
</owl:disjointWith>
```

```
</owl:Class>
<owl:Class rdf:about="# ClientInstance">
<owl:disjointWith rdf:resource="# ClientDescription"/>
<owl:disjointWith>
<owl:Class rdf:about="# ClientDescriptionSchema"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasDescription"/>
<owl:allValuesFrom rdf:resource="# ClientDescription"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="# Client"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="# NamedSecurityEnforcement">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="implements"/>
</owl:onProperty>
<owl:allValuesFrom rdf:resource="# NamedSecurityPolicy"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
```

```
<owl:Restriction>
<owl:onProperty rdf:resource="# hasDescription"/>
<owl:allValuesFrom>
<owl:Class rdf:about="# SEDescription"/>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="# SecurityEnforcement"/>
</rdfs:subClassOf>
<owl:disjointWith>
<owl:Class rdf:about="# SEDescription"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# SEDescriptionSchema"/>
</owl:Class>
<owl:Class rdf:about="# VirtualResource">
<owl:disjointWith>
<owl:Class rdf:about="# GridConnection"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# VO"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# Application"/>
<owl:disjointWith rdf:resource="# PhysicalResource"/>
<owl:disjointWith>
<owl:Class rdf:about="# Service"/>
```

```
</owl:disjointWith>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Logical representation of a/multiple physical resource(s).</rdfs:comment>
<owl:disjointWith>
<owl:Class rdf:about="# GridNode"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# GridMiddleware"/>
</owl:disjointWith>
<rdfs:subClassOf rdf:resource="# kernelOntoSkeleton"/>
<owl:disjointWith>
<owl:Class rdf:about="# Partner"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# Support"/>
<owl:disjointWith>
<owl:Class rdf:about="# QoS"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Security"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# GridJob"/>
</owl:Class>
<owl:Class rdf:about="# PRDescriptionSchema">
<rdfs:subClassOf rdf:resource="# PhysicalResource"/>
<owl:disjointWith rdf:resource="# NamedPhysicalResource"/>
<owl:disjointWith>
```

```
<owl:Class rdf:about="# PRDescription"/>
</owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="# SPDescriptionSchema">
<rdfs:subClassOf>
<owl:Class rdf:about="# SecurityPolicy"/>
</rdfs:subClassOf>
<owl:disjointWith rdf:resource="# NamedSecurityPolicy"/>
<owl:disjointWith rdf:resource="# SPDescription"/>
</owl:Class>
<owl:Class rdf:about="# GridMiddleware">
<owl:disjointWith rdf:resource="# Support"/>
<owl:disjointWith rdf:resource="# VirtualResource"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Software stacks designed to virtualise and provide access to
physical resources.</rdfs:comment>
<owl:disjointWith>
<owl:Class rdf:about="# QoS"/>
</owl:disjointWith>
<rdfs:subClassOf rdf:resource="# kernelOntoSkeleton"/>
<owl:disjointWith>
<owl:Class rdf:about="# VO"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# GridJob"/>
<owl:disjointWith>
<owl:Class rdf:about="# GridNode"/>
```

```
</owl:disjointWith>
<owl:disjointWith rdf:resource="# PhysicalResource"/>
<owl:disjointWith>
<owl:Class rdf:about="# Partner"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Security"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Service"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# Application"/>
<owl:disjointWith>
<owl:Class rdf:about="# GridConnection"/>
</owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="# Client">
<owl:disjointWith rdf:resource="# Container"/>
<rdfs:subClassOf>
<owl:Class rdf:about="# GridNode"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="# ApplicationDescriptionSchema">
<rdfs:subClassOf rdf:resource="# Application"/>
<owl:disjointWith rdf:resource="# ApplicationDescription"/>
<owl:disjointWith rdf:resource="# NamedApplication"/>
```

```
</owl:Class>
<owl:Class rdf:about="# GCDescription">
<owl:disjointWith rdf:resource="# GCDescriptionSchema"/>
<owl:disjointWith>
<owl:Class rdf:about="# GridConnectionInstance"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
<owl:allValuesFrom rdf:resource="# GCDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="# GridConnection"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="# NamedQoSEnforcement">
<rdfs:subClassOf>
<owl:Restriction>
```

```
<owl:allValuesFrom rdf:resource="# NamedQoSPolicy"/>
<owl:onProperty rdf:resource="# implements"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasDescription"/>
<owl:allValuesFrom rdf:resource="# QEDescription"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="# QoSEnforcement"/>
<owl:disjointWith rdf:resource="# QEDescription"/>
<owl:disjointWith rdf:resource="# QEDescriptionSchema"/>
</owl:Class>
<owl:Class rdf:about="# NamedGridPartner">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasDescription"/>
<owl:allValuesFrom>
<owl:Class rdf:ID="GPDescription"/>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="# GridPartner"/>
</rdfs:subClassOf>
```

```
<owl:disjointWith>
<owl:Class rdf:about="# GPDescription"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:ID="GPDescriptionSchema"/>
</owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="# QoS">
<owl:disjointWith rdf:resource="# Support"/>
<owl:disjointWith rdf:resource="# VirtualResource"/>
<rdfs:subClassOf rdf:resource="# kernelOntoSkeleton"/>
<owl:disjointWith rdf:resource="# PhysicalResource"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Quality measure and control in order to provide different
priority to different grid users, or guarantee a certain level of
performance to grid applications.</rdfs:comment>
<owl:disjointWith>
<owl:Class rdf:about="# VO"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# GridNode"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Security"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# GridMiddleware"/>
```

```
<owl:disjointWith>
<owl:Class rdf:about="# GridConnection"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# GridJob"/>
<owl:disjointWith>
<owl:Class rdf:about="# Partner"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Service"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# Application"/>
</owl:Class>
<owl:Class rdf:about="# PRDescription">
<owl:disjointWith rdf:resource="# NamedPhysicalResource"/>
<owl:disjointWith rdf:resource="# PRDescriptionSchema"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
<owl:allValuesFrom rdf:resource="# PRDescriptionSchema"/>
```

```
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="# PhysicalResource"/>
</owl:Class>
<owl:Class rdf:about="# GridConnectionInstance">
<owl:disjointWith rdf:resource="# GCDescription"/>
<owl:disjointWith rdf:resource="# GCDescriptionSchema"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="connects"/>
</owl:onProperty>
<owl:allValuesFrom>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<owl:Class rdf:about="# ContainerInstance"/>
<owl:Class rdf:about="# ClientInstance"/>
</owl:intersectionOf>
</owl:Class>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasDescription"/>
<owl:allValuesFrom rdf:resource="# GCDescription"/>
```

```
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="# GridConnection"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="# GridConnection">
<owl:disjointWith rdf:resource="# Application"/>
<owl:disjointWith>
<owl:Class rdf:about="# VO"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# Support"/>
<owl:disjointWith>
<owl:Class rdf:about="# Partner"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# GridNode"/>
</owl:disjointWith>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Logical connections between grid nodes.</rdfs:comment>
<owl:disjointWith rdf:resource="# QoS"/>
<owl:disjointWith rdf:resource="# GridJob"/>
<owl:disjointWith>
<owl:Class rdf:about="# Security"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# VirtualResource"/>
```

```
<owl:disjointWith rdf:resource="# PhysicalResource"/>
<owl:disjointWith>
<owl:Class rdf:about="# Service"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# GridMiddleware"/>
<rdfs:subClassOf rdf:resource="# kernelOntoSkeleton"/>
</owl:Class>
<owl:Class rdf:about="# Security">
<owl:disjointWith>
<owl:Class rdf:about="# GridNode"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# Partner"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# VirtualResource"/>
<owl:disjointWith rdf:resource="# GridMiddleware"/>
<owl:disjointWith rdf:resource="# Support"/>
<owl:disjointWith rdf:resource="# PhysicalResource"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
The control of risks related to the access of a VO,
a grid service, data, etc.</rdfs:comment>
<owl:disjointWith rdf:resource="# GridConnection"/>
<owl:disjointWith rdf:resource="# GridJob"/>
<owl:disjointWith rdf:resource="# Application"/>
<owl:disjointWith rdf:resource="# QoS"/>
<owl:disjointWith>
```

```
<owl:Class rdf:about="# VO"/>
</owl:disjointWith>
<rdfs:subClassOf rdf:resource="# kernelOntoSkeleton"/>
<owl:disjointWith>
<owl:Class rdf:about="# Service"/>
</owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="# Partner">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Specified role in the access to resources and/or services
within a VO or a grid application.</rdfs:comment>
<owl:disjointWith rdf:resource="# VirtualResource"/>
<owl:disjointWith>
<owl:Class rdf:about="# VO"/>
</owl:disjointWith>
<rdfs:subClassOf rdf:resource="# kernelOntoSkeleton"/>
<owl:disjointWith rdf:resource="# Support"/>
<owl:disjointWith>
<owl:Class rdf:about="# Service"/>
</owl:disjointWith>
<owl:disjointWith>
<owl:Class rdf:about="# GridNode"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# GridJob"/>
<owl:disjointWith rdf:resource="# GridMiddleware"/>
<owl:disjointWith rdf:resource="# PhysicalResource"/>
```

```
<owl:disjointWith rdf:resource="# Security"/>
<owl:disjointWith rdf:resource="# Application"/>
<owl:disjointWith rdf:resource="# QoS"/>
<owl:disjointWith rdf:resource="# GridConnection"/>
</owl:Class>
<owl:Class rdf:about="# NamedJobInstance">
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:ID="JobDescription"/>
</owl:allValuesFrom>
<owl:onProperty rdf:resource="# hasDescription"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# NamedSecurityPolicy"/>
<owl:onProperty rdf:resource="# hasSecuritypolicy"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="# GridJob"/>
<owl:disjointWith>
<owl:Class rdf:about="# JobDescription"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Restriction>
```

```
<owl:onProperty>
<owl:ObjectProperty rdf:ID="hasClient"/>
</owl:onProperty>
<owl:allValuesFrom rdf:resource="# ClientInstance"/>
</owl:Restriction>
</rdfs:subClassOf>
<owl:disjointWith>
<owl:Class rdf:ID="JobDescriptionSchema"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="hasVirtualResource"/>
</owl:onProperty>
<owl:allValuesFrom>
<owl:Class rdf:about="# NamedVirtualResource"/>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# NamedService"/>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="hasService"/>
</owl:onProperty>
</owl:Restriction>
```

```
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# NamedQoSPolicy"/>
<owl:onProperty rdf:resource="# hasQoSPolicy"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:about="# ContainerInstance"/>
</owl:allValuesFrom>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="hasContainer"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="# NamedVO">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasVirtualResource"/>
<owl:allValuesFrom>
<owl:Class rdf:about="# NamedVirtualResource"/>
</owl:allValuesFrom>
</owl:Restriction>
```

```
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasContainer"/>
<owl:allValuesFrom>
<owl:Class rdf:about="# ContainerInstance"/>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# NamedService"/>
<owl:onProperty rdf:resource="# hasService"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:about="# NamedGridMiddleware"/>
</owl:allValuesFrom>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="hasGridMiddleware"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
```

```
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# GridConnectionInstance"/>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="hasGridConnection"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasClient"/>
<owl:allValuesFrom rdf:resource="# ClientInstance"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="supportApplication"/>
</owl:onProperty>
<owl:allValuesFrom rdf:resource="# NamedApplication"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasQoSPolicy"/>
<owl:allValuesFrom rdf:resource="# NamedQoSPolicy"/>
</owl:Restriction>
```

```
</rdfs:subClassOf>
<owl:disjointWith rdf:resource="# VODescription"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# NamedSecurityPolicy"/>
<owl:onProperty rdf:resource="# hasSecuritypolicy"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasSecurityEnforcement"/>
<owl:allValuesFrom rdf:resource="# NamedSecurityEnforcement"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# NamedQoSEnforcement"/>
<owl:onProperty rdf:resource="# hasQoSEnforcement"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:about="# hasPartner"/>
</owl:onProperty>
<owl:allValuesFrom rdf:resource="# NamedGridPartner"/>
```

```
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="# VO"/>
</rdfs:subClassOf>
<owl:disjointWith>
<owl:Class rdf:about="# VODescriptionSchema"/>
</owl:disjointWith>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# VODescription"/>
<owl:onProperty rdf:resource="# hasDescription"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="# SecurityPolicy">
<rdfs:subClassOf rdf:resource="# Security"/>
<owl:disjointWith>
<owl:Class rdf:about="# SecurityEnforcement"/>
</owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="# ServiceDescriptionSchema">
<owl:disjointWith rdf:resource="# NamedService"/>
<owl:disjointWith rdf:resource="# ServiceDescription"/>
<rdfs:subClassOf>
<owl:Class rdf:about="# Service"/>
```

```
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="# JobDescription">
<owl:disjointWith>
<owl:Class rdf:about="# JobDescriptionSchema"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# NamedJobInstance"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:about="# JobDescriptionSchema"/>
</owl:allValuesFrom>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="# GridJob"/>
</owl:Class>
<owl:Class rdf:about="# ClientDescriptionSchema">
<rdfs:subClassOf rdf:resource="# Client"/>
```

```
<owl:disjointWith rdf:resource="# ClientDescription"/>
<owl:disjointWith rdf:resource="# ClientInstance"/>
</owl:Class>
<owl:Class rdf:about="# SEDescription">
<owl:disjointWith rdf:resource="# NamedSecurityEnforcement"/>
<owl:disjointWith rdf:resource="# SEDescriptionSchema"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
<owl:allValuesFrom rdf:resource="# SEDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="# SecurityEnforcement"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="# VO">
<owl:disjointWith>
<owl:Class rdf:about="# Service"/>
```

```
</owl:disjointWith>
<owl:disjointWith rdf:resource="# GridMiddleware"/>
<owl:disjointWith rdf:resource="# Application"/>
<owl:disjointWith rdf:resource="# GridJob"/>
<owl:disjointWith rdf:resource="# PhysicalResource"/>
<owl:disjointWith>
<owl:Class rdf:about="# GridNode"/>
</owl:disjointWith>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
A type of administrative domain for sharing resources across
different institutions and/or individuals in order to achieve a specific goal.
</rdfs:comment>
<rdfs:subClassOf rdf:resource="# kernelOntoSkeleton"/>
<owl:disjointWith rdf:resource="# Security"/>
<owl:disjointWith rdf:resource="# VirtualResource"/>
<owl:disjointWith rdf:resource="# Partner"/>
<owl:disjointWith rdf:resource="# GridConnection"/>
<owl:disjointWith rdf:resource="# QoS"/>
<owl:disjointWith rdf:resource="# Support"/>
</owl:Class>
<owl:Class rdf:about="# APDescriptionSchema">
<rdfs:subClassOf rdf:resource="# ApplicationPartner"/>
<owl:disjointWith rdf:resource="# APDescription"/>
<owl:disjointWith rdf:resource="# NamedApplicationPartner"/>
</owl:Class>
<owl:Class rdf:about="# ContainerInstance">
```

```
<owl:disjointWith rdf:resource="# ContainerDescription"/>
<owl:disjointWith rdf:resource="# ContainerDescriptionSchema"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasDescription"/>
<owl:allValuesFrom rdf:resource="# ContainerDescription"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="# Container"/>
</owl:Class>
<owl:Class rdf:about="# GridNode">
<owl:disjointWith rdf:resource="# GridMiddleware"/>
<owl:disjointWith rdf:resource="# Support"/>
<owl:disjointWith rdf:resource="# GridConnection"/>
<owl:disjointWith rdf:resource="# Application"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Nodes in a grid environment, including service containers,
grid clients, etc.</rdfs:comment>
<rdfs:subClassOf rdf:resource="# kernelOntoSkeleton"/>
<owl:disjointWith rdf:resource="# GridJob"/>
<owl:disjointWith rdf:resource="# VirtualResource"/>
<owl:disjointWith rdf:resource="# PhysicalResource"/>
<owl:disjointWith rdf:resource="# Partner"/>
<owl:disjointWith rdf:resource="# VO"/>
<owl:disjointWith>
<owl:Class rdf:about="# Service"/>
```

```
</owl:disjointWith>
<owl:disjointWith rdf:resource="# QoS"/>
<owl:disjointWith rdf:resource="# Security"/>
</owl:Class>
<owl:Class rdf:about="# GridPartner">
<rdfs:subClassOf rdf:resource="# Partner"/>
<owl:disjointWith rdf:resource="# ApplicationPartner"/>
</owl:Class>
<owl:Class rdf:about="# VODescriptionSchema">
<owl:disjointWith rdf:resource="# NamedVO"/>
<owl:disjointWith rdf:resource="# VODescription"/>
<rdfs:subClassOf rdf:resource="# VO"/>
</owl:Class>
<owl:Class rdf:about="# GPDescriptionSchema">
<owl:disjointWith>
<owl:Class rdf:about="# GPDescription"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="# NamedGridPartner"/>
<rdfs:subClassOf rdf:resource="# GridPartner"/>
</owl:Class>
<owl:Class rdf:about="# NamedVirtualResource">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasSecuritypolicy"/>
<owl:allValuesFrom rdf:resource="# NamedSecurityPolicy"/>
</owl:Restriction>
```

```
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# NamedQoSPolicy"/>
<owl:onProperty rdf:resource="# hasQoSPolicy"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# NamedPhysicalResource"/>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="hasPhysicalResource"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:about="# NamedGridMiddleware"/>
</owl:allValuesFrom>
<owl:onProperty rdf:resource="# hasGridMiddleware"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasDescription"/>
```

```
<owl:allValuesFrom rdf:resource="# VRDescription"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="# VirtualResource"/>
<owl:disjointWith rdf:resource="# VRDescription"/>
<owl:disjointWith rdf:resource="# VRDescriptionSchema"/>
</owl:Class>
<owl:Class rdf:about="# Service">
<owl:disjointWith rdf:resource="# GridJob"/>
<owl:disjointWith rdf:resource="# GridNode"/>
<owl:disjointWith rdf:resource="# VirtualResource"/>
<owl:disjointWith rdf:resource="# QoS"/>
<owl:disjointWith rdf:resource="# GridMiddleware"/>
<owl:disjointWith rdf:resource="# Application"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Software component, which provides platform-independent
protocols and standards used for exchanging data between applications.
It can be either stateful or stateless, as compared with a traditional
Web service.</rdfs:comment>
<rdfs:subClassOf rdf:resource="# kernelOntoSkeleton"/>
<owl:disjointWith rdf:resource="# Security"/>
<owl:disjointWith rdf:resource="# GridConnection"/>
<owl:disjointWith rdf:resource="# PhysicalResource"/>
<owl:disjointWith rdf:resource="# Partner"/>
<owl:disjointWith rdf:resource="# VO"/>
<owl:disjointWith rdf:resource="# Support"/>
```

```
</owl:Class>
<owl:Class rdf:about="# SecurityEnforcement">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
Implemented mechanisms, which realise individual or a set
of security policies.</rdfs:comment>
<owl:disjointWith rdf:resource="# SecurityPolicy"/>
<rdfs:subClassOf rdf:resource="# Security"/>
</owl:Class>
<owl:Class rdf:about="# NamedGridMiddleware">
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# NamedSecurityPolicy"/>
<owl:onProperty rdf:resource="# hasSecuritypolicy"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasSecurityEnforcement"/>
<owl:allValuesFrom rdf:resource="# NamedSecurityEnforcement"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# NamedQoSEnforcement"/>
<owl:onProperty rdf:resource="# hasQoSEnforcement"/>
</owl:Restriction>
```

```
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# hasQoSPolicy"/>
<owl:allValuesFrom rdf:resource="# NamedQoSPolicy"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# GMDescription"/>
<owl:onProperty rdf:resource="# hasDescription"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="# GridMiddleware"/>
<owl:disjointWith rdf:resource="# GMDescription"/>
<owl:disjointWith rdf:resource="# GMDescriptionSchema"/>
</owl:Class>
<owl:Class rdf:about="# GPDescription">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema# int" >
1</owl:minCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
```

```
<owl:Restriction>
<owl:allValuesFrom rdf:resource="# GPDescriptionSchema"/>
<owl:onProperty rdf:resource="# useDescriptionSchema"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="# GridPartner"/>
<owl:disjointWith rdf:resource="# GPDescriptionSchema"/>
<owl:disjointWith rdf:resource="# NamedGridPartner"/>
</owl:Class>
<owl:Class rdf:about="# JobDescriptionSchema">
<rdfs:subClassOf rdf:resource="# GridJob"/>
<owl:disjointWith rdf:resource="# JobDescription"/>
<owl:disjointWith rdf:resource="# NamedJobInstance"/>
</owl:Class>
<owl:ObjectProperty rdf:about="# hasPartner">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema# string" >
</rdfs:comment>
</owl:ObjectProperty>
<NamedApplicationPartner rdf:ID="bg_uop_008">
<hasDescription>
<APDescription rdf:ID="Des_0bgUop008_1">
<useDescriptionSchema>
<APDescriptionSchema rdf:ID="bioprofilePersonSchema"/>
</useDescriptionSchema>
</APDescription>
</hasDescription>
```

```
<isA>
<NamedGridPartner rdf:ID="patient_9817235">
<hasDescription>
<GPDescription rdf:ID="Des_0Patient9817235_1">
<useDescriptionSchema>
<GPDescriptionSchema rdf:ID="IeSEPersonSchema"/>
</useDescriptionSchema>
</GPDescription>
</hasDescription>
</NamedGridPartner>
</isA>
</NamedApplicationPartner>
<ServiceDescription rdf:ID="Des_GT4DIS_1">
<useDescriptionSchema>
<ServiceDescriptionSchema rdf:ID="_1.2"/>
</useDescriptionSchema>
</ServiceDescription>
<ServiceDescription rdf:ID="Des_UoPSIAS_001_1">
<useDescriptionSchema rdf:resource="# _1.2"/>
</ServiceDescription>
<NamedService rdf:ID="UoP_EEGMetadataAccessService_001">
<hasDescription>
<ServiceDescription rdf:ID="Des_UoPEMAS_001_1">
<useDescriptionSchema rdf:resource="# _1.2"/>
</ServiceDescription>
</hasDescription>
```

```
</NamedService>
<NamedService rdf:ID="TSI_SubInfoAccessService_127">
<hasDescription>
<ServiceDescription rdf:ID="Des_TSISIAS_127_1">
<useDescriptionSchema rdf:resource="# _1.2"/>
</ServiceDescription>
</hasDescription>
</NamedService>
<ServiceDescription rdf:ID="Des_0EMQS_89712_1">
<useDescriptionSchema rdf:resource="# _1.2"/>
</ServiceDescription>
<APDescription rdf:ID="Des_0bgUop010_1">
<useDescriptionSchema rdf:resource="# bioprofilePersonSchema"/>
</APDescription>
<APDescription rdf:ID="Des_0bgUop006_1">
<useDescriptionSchema rdf:resource="# bioprofilePersonSchema"/>
</APDescription>
<NamedService rdf:ID="GT4_DefaultIndexService">
<hasDescription rdf:resource="# Des_GT4DIS_1"/>
</NamedService>
<NamedService rdf:ID="TUT_EEGMetadataAccessService_06743">
<hasDescription>
<ServiceDescription rdf:ID="Des_TUTEMAS_06743_1">
<useDescriptionSchema rdf:resource="# _1.2"/>
</ServiceDescription>
</hasDescription>
```

```
</NamedService>
<NamedService rdf:ID="TUT_SubInfoAccessService_91286">
<hasDescription>
<ServiceDescription rdf:ID="Des_TUTSIAS_91286_1">
<useDescriptionSchema rdf:resource="# _1.2"/>
</ServiceDescription>
</hasDescription>
</NamedService>
<APDescription rdf:ID="Des_0bgUop004_1">
<useDescriptionSchema rdf:resource="# bioprofilePersonSchema"/>
</APDescription>
<NamedService rdf:ID="TSI_EEGMetadataAccessService_098">
<hasDescription>
<ServiceDescription rdf:ID="Des_TSIEMAS_098_1">
<useDescriptionSchema rdf:resource="# _1.2"/>
</ServiceDescription>
</hasDescription>
</NamedService>
<NamedApplicationPartner rdf:ID="bg_uop_004">
<hasDescription rdf:resource="# Des_0bgUop004_1"/>
<isA>
<NamedGridPartner rdf:ID="clinician_0137532">
<hasDescription>
<GPDescription rdf:ID="Des_0Clinician0137532_1">
<useDescriptionSchema rdf:resource="# IeSEPersonSchema"/>
</GPDescription>
```

```
</hasDescription>
</NamedGridPartner>
</isA>
</NamedApplicationPartner>
<ServiceDescription rdf:ID="Des_GT4MJFS_1">
<useDescriptionSchema rdf:resource="# _1.2"/>
</ServiceDescription>
<NamedApplicationPartner rdf:ID="bg_uop_006">
<isA rdf:resource="# clinician_0137532"/>
<hasDescription rdf:resource="# Des_0bgUop006_1"/>
</NamedApplicationPartner>
<NamedService rdf:ID="UoP_EEGAnalysisMetadataAccessService_001">
<hasDescription>
<ServiceDescription rdf:ID="Des_UoPEAMAS_001_1">
<useDescriptionSchema rdf:resource="# _1.2"/>
</ServiceDescription>
</hasDescription>
</NamedService>
<NamedService rdf:ID="DistributedQueryService_39458">
<requireService rdf:resource="# GT4_DefaultIndexService"/>
<hasDescription>
<ServiceDescription rdf:ID="Des_0DQS_39458_1">
<useDescriptionSchema rdf:resource="# _1.2"/>
</ServiceDescription>
</hasDescription>
</NamedService>
```

```
<NamedService rdf:ID="EEGAnalysisMetadataQueryService_12413">
<requireService rdf:resource="# GT4_DefaultIndexService"/>
<requireService rdf:resource="# DistributedQueryService_39458"/>
<hasDescription>
<ServiceDescription rdf:ID="Des_0EAMQS_12413_1">
<useDescriptionSchema rdf:resource="# _1.2"/>
</ServiceDescription>
</hasDescription>
</NamedService>
<ServiceDescription rdf:ID="Des_0PJS_23478_1">
<useDescriptionSchema rdf:resource="# _1.2"/>
</ServiceDescription>
<NamedService rdf:ID="EEGAnalysisService_01023">
<hasDescription>
<ServiceDescription rdf:ID="Des_0EAS_01023_1">
<useDescriptionSchema rdf:resource="# _1.2"/>
</ServiceDescription>
</hasDescription>
<requireService rdf:resource="# GT4_DefaultIndexService"/>
<requireService>
<NamedService rdf:ID="EEGMetadataQueryService_89712">
<hasDescription rdf:resource="# Des_0EMQS_89712_1"/>
<requireService rdf:resource="# GT4_DefaultIndexService"/>
<requireService rdf:resource="# DistributedQueryService_39458"/>
</NamedService>
</requireService>
```

```
<requireService rdf:resource="# EEGAnalysisMetadataQueryService_12413"/>
</NamedService>
<NamedService rdf:ID="GT4_ManagedJobFactoryService">
<hasDescription rdf:resource="# Des_GT4MJFS_1"/>
</NamedService>
<GPDescription rdf:ID="Des_0BioprofileResearcher_2734623">
<useDescriptionSchema rdf:resource="# IeSEPersonSchema"/>
</GPDescription>
<APDescription rdf:ID="Des_0bgUop003_1">
<useDescriptionSchema rdf:resource="# bioprofilePersonSchema"/>
</APDescription>
<GPDescription rdf:ID="Des_0Patient1273691_1">
<useDescriptionSchema rdf:resource="# IeSEPersonSchema"/>
</GPDescription>
<NamedService rdf:ID="UoP_SubInfoAccessService_001">
<hasDescription rdf:resource="# Des_UoPSIAS_001_1"/>
</NamedService>
<NamedApplicationPartner rdf:ID="bg_uop_002">
<hasDescription>
<APDescription rdf:ID="Des_0bgUop002_1">
<useDescriptionSchema rdf:resource="# bioprofilePersonSchema"/>
</APDescription>
</hasDescription>
<isA>
<NamedGridPartner rdf:ID="bioprofileResearcher_2734623">
<hasDescription rdf:resource="# Des_0BioprofileResearcher_2734623"/>
```

```
</NamedGridPartner>
</isA>
</NamedApplicationPartner>
<NamedApplicationPartner rdf:ID="bg_uop_007">
<isA rdf:resource="# clinician_0137532"/>
<hasDescription>
<APDescription rdf:ID="Des_0bgUop007_1">
<useDescriptionSchema rdf:resource="# bioprofilePersonSchema"/>
</APDescription>
</hasDescription>
</NamedApplicationPartner>
<NamedApplicationPartner rdf:ID="bg_uop_009">
<isA rdf:resource="# bioprofileResearcher_2734623"/>
<hasDescription>
<APDescription rdf:ID="Des_0bgUop009_1">
<useDescriptionSchema rdf:resource="# bioprofilePersonSchema"/>
</APDescription>
</hasDescription>
</NamedApplicationPartner>
<ServiceDescription rdf:ID="Des_MMAS_27962_1">
<useDescriptionSchema rdf:resource="# _1.2"/>
</ServiceDescription>
<ServiceDescription rdf:ID="Des_0SIQS_26578_1">
<useDescriptionSchema rdf:resource="# _1.2"/>
</ServiceDescription>
<APDescription rdf:ID="Des_0bgUop005_1">
```

```
<useDescriptionSchema rdf:resource="# bioprofilePersonSchema"/>
</APDescription>
<NamedService rdf:ID="MetadataMappingAccessService_27962">
<hasDescription rdf:resource="# Des_MMAS_27962_1"/>
</NamedService>
<NamedService rdf:ID="ParallelJobService_23478">
<hasDescription rdf:resource="# Des_0PJS_23478_1"/>
<requireService rdf:resource="# GT4_DefaultIndexService"/>
<requireService rdf:resource="# GT4_ManagedJobFactoryService"/>
</NamedService>
<ApplicationDescriptionSchema rdf:ID="KWFGridAppOnto"/>
<NamedApplicationPartner rdf:ID="bg_uop_001">
<isA rdf:resource="# clinician_0137532"/>
<hasDescription>
<APDescription rdf:ID="Des_0bgUop001_1">
<useDescriptionSchema rdf:resource="# bioprofilePersonSchema"/>
</APDescription>
</hasDescription>
</NamedApplicationPartner>
<NamedApplication rdf:ID="EEGAanlysisForDementiaDetection">
<hasPartner rdf:resource="# bg_uop_001"/>
<hasPartner rdf:resource="# bg_uop_007"/>
<hasPartner rdf:resource="# bg_uop_006"/>
<hasPartner>
<NamedApplicationPartner rdf:ID="bg_uop_005">
<isA>
```

```
<NamedGridPartner rdf:ID="patient_1273691">
<hasDescription rdf:resource="# Des_0Patient1273691_1"/>
</NamedGridPartner>
</isA>
<hasDescription rdf:resource="# Des_0bgUop005_1"/>
</NamedApplicationPartner>
</hasPartner>
<hasDescription>
<ApplicationDescription rdf:ID="Des_0EEGAanlysisForDementiaDetection_1">
<useDescriptionSchema rdf:resource="# KWFGridAppOnto"/>
</ApplicationDescription>
</hasDescription>
<requireService>
<NamedService rdf:ID="SubjectInfoQueryService_26578">
<requireService rdf:resource="# GT4_DefaultIndexService"/>
<requireService rdf:resource="# DistributedQueryService_39458"/>
<hasDescription rdf:resource="# Des_0SIQS_26578_1"/>
</NamedService>
</requireService>
<hasPartner rdf:resource="# bg_uop_009"/>
<requireService rdf:resource="# EEGAnalysisService_01023"/>
<hasPartner rdf:resource="# bg_uop_004"/>
<hasPartner rdf:resource="# bg_uop_008"/>
<hasPartner>
<NamedApplicationPartner rdf:ID="bg_uop_010">
<isA rdf:resource="# bioprofileResearcher_2734623"/>
```

```
<hasDescription rdf:resource="# Des_0bgUop010_1"/>
</NamedApplicationPartner>
</hasPartner>
<hasPartner rdf:resource="# bg_uop_002"/>
<hasPartner>
<NamedApplicationPartner rdf:ID="bg_uop_003">
<hasDescription rdf:resource="# Des_0bgUop003_1"/>
<isA rdf:resource="# clinician_0137532"/>
</NamedApplicationPartner>
</hasPartner>
</NamedApplication>
</rdf:RDF>
<!- Created with Protege (with OWL Plugin 3.3.1, Build 430)
http://protege.stanford.edu ->
```