A Metadata Service for Service Oriented Architectures

Noel King

Bachelor of Science in Computer Applications (Software Engineering)

A dissertation submitted in partial fulfilment of the

requirements for the award of

Master of Science in Computing Applications

to the

Dublin City University

School of Computing

Supervisor: Dr. Mark Roantree

September, 2005

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Masters of Science in Computer Applications is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed Noel King

Student ID 99553996

Date September, 2005

Acknowledgments

I would like to take this opportunity to thank my supervisor Dr. Mark Roantree, whose support and encouragement since starting this program has been amazing and really appreciated. Working within the Interoperable Systems Group (ISG) has been a great experience and the support from all team members has played a big role in the completion of this thesis. Martin for all our inspiring and sometimes heated discussions. Dalen for your guidance and the sharing of your vast knowledge. Zohra for your interest, ideas and support for all the work we completed here. Seamus Murphy for your year in ISG, the good fun and laid back approach. Other Bay F members who deserve a mention include Ciaran Ferry for the good laughs, proof reading and discussions we enjoyed and not to forget Caroline and Gavin for your encouragement and assistance.

My family and friends have supported me throughout this Masters and I would like to detail how grateful I am for their inspiration and help. Pauline my lovely girlfriend, you have been there during all the tough times, taken my moans and cheered me up again, I really appreciate everything you have done for me. Mam, Dad, Sean and Eoin thanks for being there to support the thesis cause. Finally Will, Fiachra and the rest of the lads for not letting me stay in on a Saturday night, emphasising the importance of enjoying life.

Abstract

Service oriented architectures provide a modern paradigm for web services allowing seamless interoperation among network applications and supporting a flexible approach to building large complex information systems. A number of industrial standards have emerged to exploit this paradigm with the development of the J2EE and .NET infrastructure platforms, communication protocol SOAP, description language WSDL and orchestration languages BPEL, XLANG and WSCI. At the same time the Semantic Web enables automated use of ontologies to describe web services in a machine interpretable language. To enable process composition and large scale resource integration over heterogeneous sources a new research initiative is needed. Current initiatives have identified the role of Peer-to-Peer networks and Service Oriented Architectures to enable large scale resource communication and integration. However this approach neglects to identify or utilise the role of Semantic Web technologies to promote greater automation and reliability using service semantics, thus a new framework is required adopting Peer-to-Peer networks, Service Oriented Architectures and Semantic Web technologies. In this context, this thesis presents a management and storage framework for a distributed service repository over a super peer network to facilitate process composition.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There is little doubt that Enterprise Software has revolutionised information systems, creating sustainable improvements in organisational efficiency and agility. Enterprise Software provides software components tightly coupled with organisational processes and model, to create enterprise across departments and external business relationships. Organisations have become to realise the need for Enterprise Software to enable their complex diverse systems communicate with each other to maximise enterprise benefits. Traditionally Enterprise Software is not an isolated system, but rather a large number of systems supporting complex cross-dependencies that have grown over many years to cause high levels of heterogeneity and redundancy. This is mainly due to the conflicting and unclear requirements of the organisation during the software design stage and the permanent changes of business dynamics, requiring new efficiencies and processes to be incorporated in the existing software. Consequently software architects are now confronted with many challenges when refactoring existing software to expand functionality while striving to reduce complexity and increase agility. This identifies a need for a strong enterprise architecture to address these structural problems.

Due to the size and apparent complexity of business models enterprise software is faced with two key challenges. Firstly, the composition of any software architecture for Enterprise Software is orthogonal to information exchange issues. This requires enterprise systems to provide efficient and accurate information communication between enterprise components to meet an organisational goal. Secondly, as a consequence of cross departmen-

tal and distributed requirements when developing Enterprise Software, developers are also faced with communication challenges outside the realm of object oriented and functional techniques. To support access to programs on remote machines, distributed computing techniques have evolved to provide a seamless and controlled mechanism for remote object invocation. This leads to the problem of integrating data between distributed heterogeneous components. Contrary to traditional access through object method parameters on isolated systems, distributed enterprise solutions require strong robust systems that overcome heterogeneous components and offer scalability, reliability and availability.

Outline. This chapter is organised as follows: the research background and motivation will be presented in §1.1, before introducing the issues addressed by this research in §1.2 with a research scenario detailed in §1.3. An overview of service oriented enterprise architecture is outlined in §1.4. This chapter concludes with the formation of a hypothesis in §1.5 and chapter results in §1.6.

1.1 Background and Motivation

This research forms part of the XQuery for Large Scale Integration Methods (XLIM) project, which focuses on the integration of both data and resources, providing full interoperability across enterprise boundaries [Kin05]. The project aims to extend the functionality offered by the XPeer Architecture [RB04] devised by Dublin City University and the University of Montpellier to create an Enterprise Software solution to overcome the issues associated with a large number of distributed heterogeneous data sources [Kin05]. XPeer focused on solving distributed issues associated with data stored at multiple sources and owned by different organisations. To address the querying of this data on a large scale overcoming heterogeneous formats, interfaces and semantics, XPeer introduced the need for a metadata service to support mediation. XPeer also addressed the problems associated with distributed data sources and deployed a wide area information management architecture to overcome these problems.

With respect to this research, a *resource* refers to a service which a business entity provides to meet a user requirement and a *process* depicts two or more resources integrating

to meet a user goal. Although XPeer enabled large scale data integration of heterogeneous data sources, this research explores the possibilities of querying distributed resources and data to provide users with their required services. In addition, the XLIM project investigates the potential of providing an operational system allowing resource discovery and integration with millions of users and thousands of business entities to support the Bologna Declaration. In June 1999, 29 European countries signed the Bologna Declaration [Eur99] to create 'A European Higher Education Area' with the following aims:

- To promote mobility of students and academic staff.
- To provide lifelong learning participation.

To achieve a European framework supporting large scale data and resource integration is mandatory to overcome the heterogeneous nature of countries and academic institutions. Current research concludes that to achieve integration of these heterogeneous systems, semantic integration is required as detailed in [DRR+03]. However, this focus has shifted with the proliferation of semantic web services. This shift has supported the emergence of many current semantic web technologies to support semantic integration and evolved to support full automation.

Motivation Using semantic web technologies to describe resources in a large scale architecture requires efficient data access for service discovery. Research analysing data storage requirements of semantic web technologies has focused on providing storage models and query languages to enable quick access to semantic web data. The RDFSuite described in [ACP+01a, ACP+01b], details a suite of tools to manage RDF data representing Internet resources. The suite is composed of a Validating RDF Parser, RDF Schema Specific Database and RDF Query language. The Validating RDF Parser analyses and validates descriptions being stored in the database. The RDF Schema Specific Database stores RDF descriptions in an ODBMS adopting a schema generation approach for representing resource descriptions and schemas as triples. To query RDF data, the RQL query language described in [KMP+03, KAP+02] adopts path expressions to support the navigation of schemas and data. This approach offers a persistent storage model for RDF data and

Schemas, while providing a rich query language in RQL to query stored data. This model's results justifies the requirement for storage of semantic web metadata to describe Internet resources. The success of storing RDF data to describe web resources, has motivated this research to provide a storage model for OWL-S, BPEL and WSDL metadata to describe web service resources in the XLIM architecture.

Our motivation is to provide a metadata service to address the large-scale resource issue, and to exploit current semantic web service technologies to provide smart discovery and automatic integration of resources and data. In this respect a metadata service with an accompanying query service is essential to allow scalability, quality, dependability and control over distributed business entities.

1.2 Issues

Consider the resource scenario created by the Declaration. For successful implementation, each institution must be involved in the enterprise software solution. Thus, this system must maintain the integrity of the Declaration required services and provide a generic adapter for existing services. From a user view point, services must be easily discovered and interacted with. In the situation where a service is not available for the user, a process should be composed where possible to meet their requirements. A new process is now composed from existing services distributed throughout Europe as discussed in §1.3. To compose any new process, existing services must be discovered using a query service, while a metadata service provides the necessary service metadata semantics for service discovery and composition. The issues associated with distributed resource integration to meet user requirements will now be discussed.

The differences between isolated centralised systems and distributed systems are significant. Here Enterprise systems are dispersed throughout Europe and maintained by their owners. The goal of these systems is to yield resources with an efficient resource metadata retrieval mechanism, responding to user resource queries. A distributed enterprise solution offers significant advantages when compared to isolated systems, thus allowing efficient metadata retrieval; higher application performance as tasks are executed in parallel

across multiple servers; supporting the clustering of applications and servers resulting in higher reliability and availability; ensuring scalability by deploying reusable components on powerful servers; and promoting reuse by allowing any distributed component access the objects functionality. This requires a distributed database solution, for storage and retrieval of resource metadata.

When registering a service in the architecture, providing usable discovery and invocation metadata is imperative to provide accurate results for user queries. Considering the complexity and diverse nature of services provided by European institutions, keyword discovery for services provided by UDDI[OAS04] is inadequate to provide reasonable understanding of the service goal or functionality. Thus descriptive and interpretable metadata is required to determine service functionality and to analyse its potential to match to the user resource query.

Resource interaction is reliant on data exchange to achieve accurate results. Although resource inputs are predefined structured data the semantic meaning of this data remains unknown. To implement a European enterprise system, resource requests and interaction must overcome language and terminology concerns. Metadata representing language and terminology must be used to provide a standard interpretation for all systems. A structured metadata technology should be used to provide system developers with a standard format allowing for data manipulation to achieve accurate understanding.

Although language and terminology can offer data interpretation, this alone can not achieve accurate mediation between resources to create a new process. Many resources can offer similar functionality, but due to other factors mediation between two services may not be possible. As a result of this, metadata is required to describe input, output, effects, precondition and properties of services to support mediation with other services.

1.3 Research Scenario

Consider the problem scenario created by the Bologna Declaration. Before entering third level education, students must build their own program of study across the entire range of European institutes. Each year comprises of two distinct semesters, and each semester

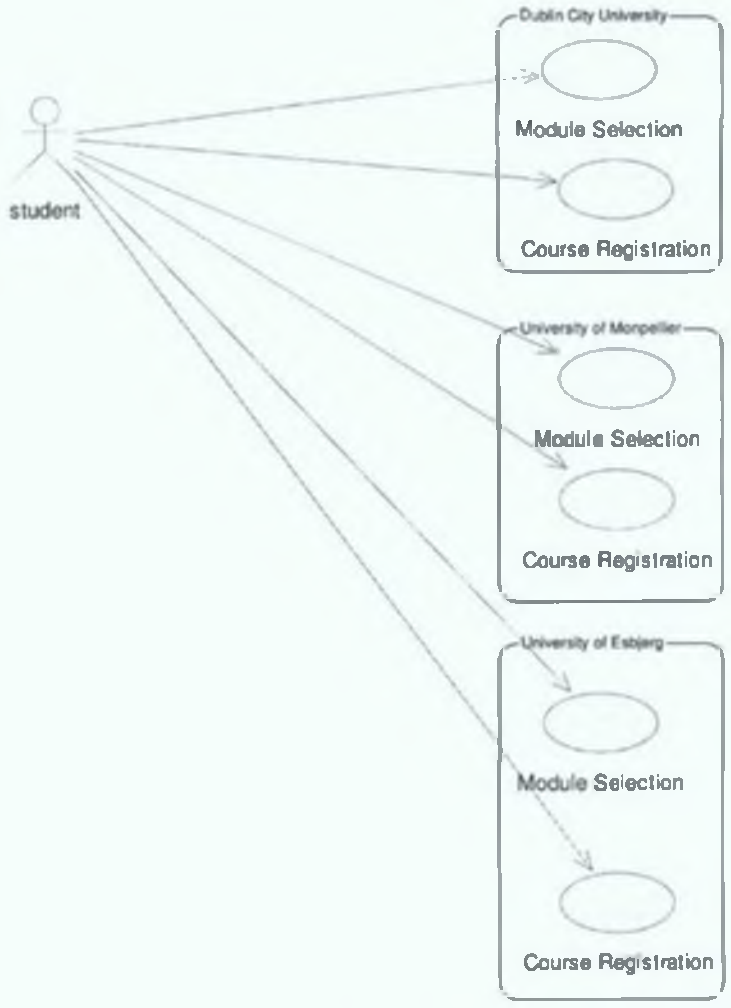


Figure 1.1: Bologna Declaration Use Case

could (in theory) be taken at a different institute. They may choose to spend the first semester in Dublin, the following three semesters at the University of Montpellier in France and year 3 in University of Esbjerg Denmark. While this offers a new level in the learning experience (and arguably one that greatly benefits the student), it presents serious technical and pragmatic issues for the institutions and their countries, not least because of the semantics involved in "course creation". To illustrate the aims of this research a simple Bologna Declaration use case example is displayed in *figure 1.1*. Rather than requiring a student to interact with all these services individually a single process interface should be provided to interact with all services on the students behalf. Due to the challenges and complexity of composing a new process there is a number of tasks that must be completed

as detailed in *figure 1.2*. Firstly, all required services must be discovered. Secondly, integration of the discovered services is achieved through negotiation and finally a process composition view is created and registered, thus providing the user with a single process interface.

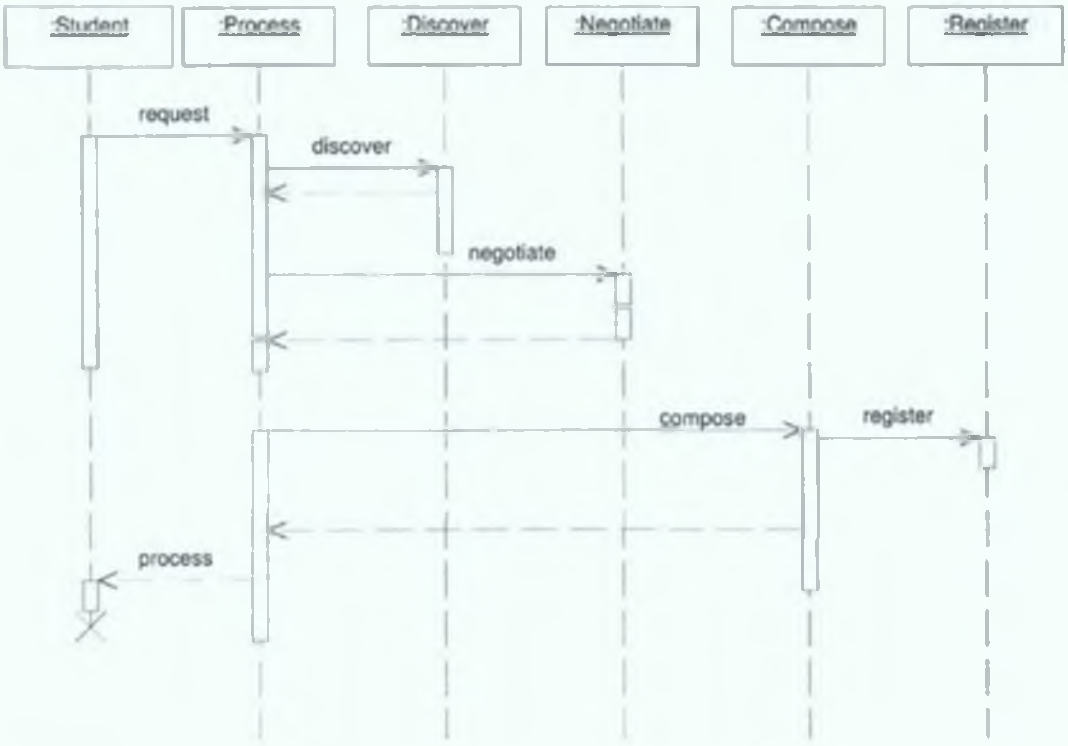


Figure 1.2: Bologna Declaration Process Composition Sequence Diagram

This research scenario offers a number of implementation challenges which must be addressed by this research.

1. All services must be semantically described to ensure accurate discovery.
2. Integration between services should be possible to build new processes.
3. Views are required of composed processes.
4. Support automatic discovery, execution and discovery where possible.
5. Quick and efficient access to all service metadata is imperative for performance and ultimately user acceptance.

This research will focus on providing the efficient access to service metadata which supports the automatic discovery and composition of this process. It is important to understand that although automation is possible with the support of metadata, ensuring process validity remains a challenge outside the scope of this research. To ensure process integrity a rule service is required to validate all composed processes and although service rules are included in the metadata to achieve service integrity, they may not contain sufficient integrity to ensure process validity.

1.4 Service Oriented Architecture

Before any metadata technologies can be investigated, distributed enterprise models must be evaluated to meet the scalable, robust and reliable requirements. The evolution of service architectures has offered a new paradigm for enterprise software to provide seamless interoperation among network applications and supporting a flexible approach to building large complex information systems. Service computing concepts have existed for many years, promoting component based and object oriented development to provide business functionality over distributed computing platforms. The evolution to Service Oriented Architectures (SOA) has revolutionised software by providing powerful tools and methodologies for software development and maintenance. An SOA is described as a software architecture that is based on the key concepts of an application front-end service, services repository, and service bus[KBS04]. The idea of SOA computing is to support an unambiguous, technology-independent, enterprise-wide standard architecture allowing communication between software modules providing application heterogeneity. Services act as an integral component in organisational layers to reduce coupling, support integration, maintain data integrity and mediate technological gaps in business functions. In order to meet this requirement and provide improved agility and efficiency SOA support simplicity, flexibility, maintainability, re-usability and decoupling of functionality and technology.

Many distributed architectures have been proposed to support the dispersed enterprise software components. However, heterogeneity remains the greatest challenge to developers, implementing distributed concepts having to overcome three core issues of communication,

system and additional runtime incompatibilities. Web Services have evolved to meet these challenges allowing easy integration over the XML data format.

1.4.1 Web Services

Web services provide a modern paradigm for distributed computing allowing seamless interoperation among network applications and supporting a flexible approach to building large complex information systems. Web services are software components available over the Internet or networks delivered using Internet technologies, as detailed in [NSS03]. The adoption of the Internet to support business-to-business (B2B) and business-to-customer (B2C) operations has been the key enabler in the web service evolution. Web services use industry standards such as XML [BPSM+04], WSDL [CGM+], SOAP [GHM+03] and UDDI [BCE+02] to encapsulate applications and publish them as services. XML based communication between service and clients supports an extremely flexible approach to integration, facilitating many of the modern day industry requirements including Enterprise Application Integration, B2B and application-to-application integration across organisational and industry boundaries.

Web services provide the necessary framework [BHM+04] to enable business interaction over a lightweight infrastructure. Unlike previous distributed architectures web services provide a number of characteristics to overcome many of their disadvantages. Messaging between service and client is based on XML data exchange, allowing for easier client integration. This is further enhanced as XML messaging is semi-structured with the support of XML Schema [W3C04]. Supporting business applications over the Internet allows for cross-platform interaction. Web services can be easily developed in any programming language including Java, C++ and C. As loose coupling is promoted, it allows components to be exposed providing unique functionality, which can be easily discovered in a registry or UDDI. The use of Internet protocols allows for easy access through corporate firewalls. Web service access is not restricted and can be invoked by many types of clients. These characteristics support an interoperable distributed environment allowing communication between platforms encapsulating any heterogeneous application.

The web service operational model can be conceptualised into three roles: service requester,

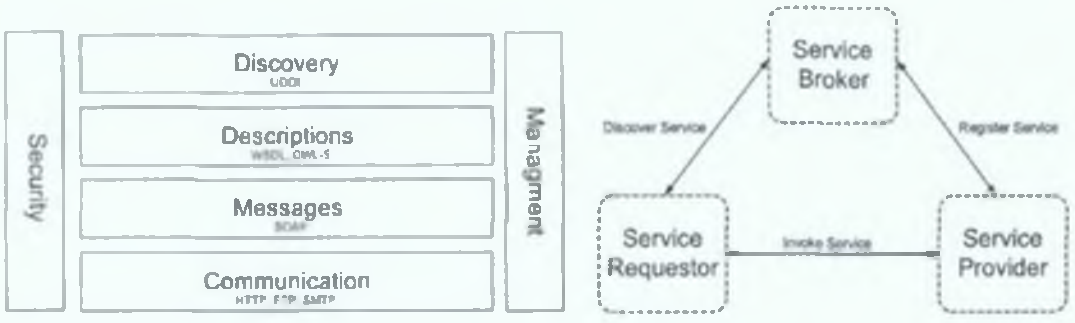


Figure 1.3: Web Service Architecture Stack and Operational Model

service broker and service provider communicating to support service invocation. As detailed in *figure 1.3*, the service provider deploys a service and registers it with the service broker. The service broker supports discovery and is queried by the service requester, who retrieves a provider interface. With the provider interface the service requester can then invoke the service provider. To support this operational model the Web Service Architecture Stack displayed in *figure 1.3* provides the necessary layers architecture for service invocation which will now be discussed.

Communication. The bottom layer is responsible for communication between distributed web service components and clients. All web services must be network enabled to allow service requesters invoke their services. Web service communication is deployed using the common Internet protocols and supports HTTP, SMTP and FTP. Although to support unambiguous communication HTTP is the de facto protocol for web service communication.

Messages. The messaging layer supports XML messaging protocols, to allow data transfer between clients and web services. Messaging requests and responses are represented using XML syntax, with Simple Object Access Protocol (SOAP) providing the industry standard for messaging. SOAP is built upon the transport layer to allow easy interaction with transport protocols while supporting publishing, binding and searching operations. The SOAP envelope structure presents a number of characteristics benefiting XML messaging offering document-centric messaging, remote procedure calls and headers allowing

orthogonal extensions to be easily included.

Descriptions. The description layers supplies the necessary semantics to invoke a web service. This primarily acts as a data layer describing a web service, which the service provider has supplied. Web Service Description Language (WSDL) is the de facto standard providing a structured XML-based service description. WSDL defines the interface and protocols for service interaction. Although WSDL provides the necessary interface for web service interaction, additional descriptions have been sought to provide machine interpretable understanding supporting automation with web services. OWL-S has emerged to meet this goal, providing intelligent descriptions incorporating WSDL. These technologies will be further discussed in *chapter* 3.

Discovery. To support resource querying, service providers register or publish descriptions with the discovery layer. Service requesters rely on the discovery layer to match functional requirements with available services. The discovery layer usually contains a registry of available services using a standard discovery mechanism such as UDDI. To enable discovery the service provider supplies WSDL information, but will complement this description with business context data, required by UDDI. Although UDDI supplies a proven interface for service registration and discovery, its ambiguous service semantics have led to the implementation of Ontology based registries which will be further discussed in *chapter* 3

Security. Security is a huge concern for any organisation implementing a large Service Oriented Architecture. The Web Service Stack identifies this need for a secure architecture to be implemented at each layer to maintain data and service integrity. In respect to this research security will not be discussed, as it lies outside the scope of the metadata service.

Management. Management of web services is concerned with supplying a set of capabilities enabling monitoring, controlling and reporting of quality and service usage. To support web service management, service providers must expose management capabilities when developing services allowing a manager audit, record and alter service characteristics.

Currently there are no de facto standards or frameworks for the web service management, so in *chapter 3* we discuss our management framework.

Where previous distributed technologies failed, the web service architecture offers a simple mechanism for applications to become distributed services accessible by anyone or any device. By promoting industry standards and non proprietary software the web service architecture reduces training and deployment costs. Using XML and supporting technologies enabling integration has allowed industry-wide connectivity over the Internet and has created a new level of organisational flexibility and agility. Implementing loosely coupled services has promoted reusable engineering and increased business interactions. Organisational service registries have supplied developers with a pool of available functionality reducing development. Web services have allowed easy collaboration with existing applications encapsulated with web service interfaces. Web services offer organisations many benefits to support existing and develop new enterprise software solutions. The evolution of web services has now introduced flow mechanisms supporting process composition. The use Ontologies has also emerged to promote automation of business processes. The web service architecture provides a complete architecture to support large scale resource integration supporting Internet based technologies, increased scalability and easier implementations. This research will now focus on the integration of web service resources.

1.5 Research Objectives

XLIM is primarily focused on the integration of heterogeneous resources supplied by European institutions, allowing students to query and invoke services on a European scale, while hiding interoperability and heterogeneity issues. XLIM extends the research conducted for the XPeer architecture that focused on data integration in a large scale scenario to investigate how services can support interoperation and data integration. The evolution of web services to support workflow and Ontologies has created a new level of machine understanding and automation, motivating our research objective. The hypothesis is that by exploiting workflow and Ontology based technologies in a metadata service, it will allow students and researchers to discover and compose new services to meet their functional

requirements. Specifically, one can easily create a programme of study for the scenario provided in §1.5 with the support of existing institution services. The contribution illustrated in this research will address metadata storage and access issues to achieve successful distributed resource integration:

Integration. Communication between services in a heterogeneous environment creates performance and semantic challenges for mediators deployed to integrate between services. XLIM supports semantically enhanced metadata in a metamodel to provide efficient access to machine processable metadata and supports the creation of integration metadata views using workflow metadata.

Process Management. Automatic process composition is crucial to match individual requirements. Machine interpretable metadata is needed to describe processes in our architecture allowing easy process composition, execution and management.

Efficient Resource Querying. Service metadata is imperative for querying resources and allowing for generic querying and the construction of global schemas. The metadata must support the easy integration enabling process composition. However, to provide efficient access to metadata in a large finite architecture, a metadata service must provide a metadata storage model to support user requirements.

Process Execution. Although metadata provides the data crucial for process composition, a composed process must be easily registered in the architecture for user invocation.

1.6 Conclusions

In this chapter the role of enterprise software was introduced outlining its significance in supporting organisational practices. However, the success of enterprise software has been limited by overly complex architectures and integration concerns. A new evolution in service architectures has emerged to allow organisation wide integration in a reliable and secure environment. Our research extended the XPeer work [RB04], which detailed

an architecture for supporting global data integration to enable automated global resource integration and composition. Furthermore, it was necessary to determine the most appropriate service oriented technology to support distributed resources required by our European context. Web services were selected to provide a flexible and scalable option, while supporting easier integration with its support for XML. However providing a European resource integration introduces many obstacles that will be addressed by this research.

The contribution of this research is to provide a metadata service addressing the issues outlined in §1.2, to allow for resource querying and integration. The first stage in resolving these issues is the adoption of the service architecture and associated metadata technologies which is detailed in *chapter 3*. The XLIM Metadata Service supporting the XLIM architecture will be presented in *chapter 4*. While in *chapter 5* the Metadata Management Framework E-business layer will be discussed, detailing the querying and access requirements of metadata to support resource discovery and integration. Finally in *chapter 6* research conclusions and potential areas for further research are discussed.

Chapter 2

Related Research

The previous chapter motivated the need to create a metadata service supporting large scale resource discovery and integration, using the XPeer super peer architecture as a basis for this work. XPeer demonstrated the benefits of utilising a super peer architecture for data integration. To realise these benefits, our hypothesis focused on resource metadata facilitating global component interaction, enabled by web service technologies. Although web services provide the communication methodology, a powerful metadata service is required to realise the full potential of resource integration. With the emergence of many semantic and workflow metadata languages, there has been much focus on developing architectures that can apply these metadata technologies to support automatic discovery of services and composition of processes. However there are many web service and distributed technologies currently available and it is necessary to identify an ideal metadata mix for the large quantity of resources available in institutions. In this chapter similar projects which influenced this research are analysed, these projects were evaluate on the following criteria:

1. Technologies adopted.
2. Benefits offered by the technologies chosen.
3. Limitation of technologies.
4. Relevance of technologies to support XLIM requirements.

This chapter is structured as follows: in §2.1 to §2.3 analysis of similar projects is conducted, while §2.5 presents our conclusions.

2.1 OntoServ Project

After identifying the opportunities of emerging web service and distributed technologies to allow effective knowledge management across industry enterprises, the Industrial Ontologies Research Group proposed the OntoServ [TK03, KKT+04] environment. The focus of this work is to provide an E-business infrastructure to enable integration between organisation's information systems to support full machine collaboration with all stakeholders. With the recent trend of organisations requiring industry wide integration, research has focused on a combination of Semantic Web and Peer to Peer research to create powerful and robust features [FvHK+00]. Using the proliferation of web based resources to support distributed computing, OntoServ presents a global P2P and E-business infrastructure to automate integration of enterprise data and resources. This framework offers robustness and flexibility in a dynamic environment using existing technologies, which will now be discussed in the OntoServ context.

In creating a robust environment for organisational collaboration, OntoServ brought together distributed technologies with web service and semantic web metadata. By using a web service interface for OntoServ components, it facilitated a reliable platform offering flexible integration and resource interoperability. To allow for the automated management of industrial resources, OntoServ adopted Ontologies namely the Ontology Web Language [SWM04]. Ontologies are used to provide machine understanding for knowledge based data, available resources and taxonomies of services and to support intelligent query routing over OntoServ. DAML-S (now OWL-S [MBH+04]) provides the basis of resource metadata in this environment to enable intelligent resource discovery and invocation. As a consequence of industry wide collaboration, OntoServ deployed a Peer-to-Peer global network to increase metadata discovery efficiency and facilitate resource integration through their peer model OntoShell.

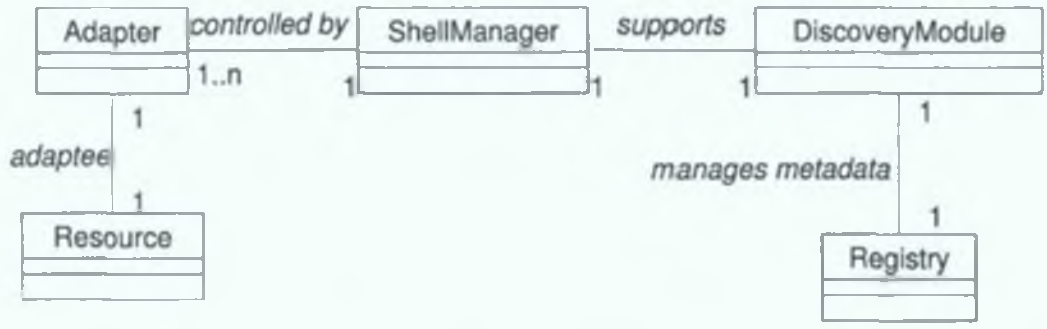


Figure 2.1: The OntoShell Model

2.1.1 OntoShell Model Overview

OntoShell displayed in figure 2.1, is the core component which utilises OntoServ metadata to support their integration aims. OntoShells are distributed throughout the OntoServ global peer network facilitating the inclusion of resources into the architecture using their resource metadata technologies. This model offers two key functions to the architecture, it supports the inclusion of resources into the architecture through adapters, representing their functionality in their registry and aggregates resource metadata representations for efficient querying strategies.

Resource. A resource is any component available for use with the OntoServ environment. This resource can be software, hardware or any human interaction required to support collaboration. Resources are accessed as services by other OntoServ resources and are described using Ontologies and DAML-S metadata.

Adapter. This adapts legacy software and resources to the Semantic Web service environment. It provides a generic-adapter component to allow mediation between the service-specific data and the communication and protocols of the existing resource. The adapter acts as the key component allowing all industry resources access the OntoServ environment.

ShellManager. The `ShellManager` processes all logic for the OntoShell, to allow for the integration of the different services and customers with the shell. All the OntoShell

services are managed by the `ShellManager` and resource metadata is stored in a registry to facilitate easy discovery.

DiscoveryModule and Registry. To allow other OntoShell resources within the OntoServ environment discover services, each shell provides a `DiscoveryModule` with associated registry of resources. All shell resources are registered in the `DiscoveryModule` by the `ShellManager` and all resource metadata including Ontologies are stored in the registry. The `DiscoveryModule` supports a standard discovery interface to allow querying of its registry.

2.1.2 P2P Network and Resource Formation

For efficient discovery and composition of services, the OntoServ framework adopts the P2P structure with Ontology metadata for network formation purposes. Instead of adopting a centralised approach OntoServ investigated the potential of forming partner relationships between providers through their OntoShell model. Their P2P model is based on a generalised profile approach for peers with relationships between partners allowing any node on the P2P network answer queries for other nodes. To compose partnerships between generalisations, OntoServ specifies a tree structured representation of Ontology classes to define these relationships. If one considers a community to be a set of related partners, by adopting the OntoServ model, using the generalised approach allows all members of that community to answer queries for the community.

When creating P2P clusters for service composition OntoServ identifies two goals that must be considered. Firstly, the cooperation of different service provider types to compose a new service and secondly, the cooperation of similar providers for collaboration between resources. In respect to a process composed from different service providers an Ontology is required to describe the three layers available to create the new process. This tree structured Ontology class representation displays three layers of abstraction using the *part-of* relationships with atomic services available at the bottom layer. The middle layer contains compound services clustering similar service provider types together. While the top layer is composed of processes, providing the access point for users and cannot be used for the

composition of higher levels. The primary objective of Ontology based representation is to improve discovery and composition of services based on similar providers.

OntoServ also adopts the clustering of services according to similarities into a tree based ontology structure, however in this situation the ontology class holds a *subclass-Of* relationship with the other layers. For purposes of this architecture there are four ontology classifications: Location, Methods, Service Quality and Service Costs. As a result Ontologies are composed based on these classifications to determine and enforce similarities between providers. This allows service providers in OntoServ or actual OntoShells specify their relevant Ontologies to combine similar services into the same Ontology, while allowing users choose the Ontology they join, there is a requirement to provide some validation or rules to ensure reliability.

When providing service Ontology classes there is a need to ensure accurate service representation to reduce negative user interaction caused by over generalised or too specific descriptions. OntoServ provides a meeting platform for service agents to interact to find partners. In this environment similar services interact and analyse similarity based on the four Ontology classification detailed in the previous paragraph. The formation of the community will evolve from the generalised profile of its members. As some agents may "lie" about the services they provide causing the service users to lose trust in the cluster, the OntoShell may be used to ensure validity. In this scenario the OntoShell acts as a controller responsible for registering child OntoShells, validating them and refusing negative results from joining the community. The mother shells are responsible for the child service advertising, grouping and mediation to support efficient search and communication. This model provides a centralised approach for access and management of communities based on profiles to enhance queries and composition.

2.1.3 Limitations

To summarise, while OntoServ offers a reliable and scalable architecture supported by DAML-S to enable the automatic discovery and composition of services, there are still some issues with respect to the metadata technologies and functionality provided. Although this project offers a similar motivation, by not utilising a metadata service it

cannot provide the same level of functionality. This work details the role of Ontologies in building communities of profiles over a P2P network, with communities containing resources described using DAML-S (now OWL-S) upper Ontology. When discussing these two metadata technologies, OntoServ neglects to identify the relationship between Ontology profiles of services and the DAML-S service category and type components. OntoServ discusses the role of an integration module to provide mediation between services in a composed process, but neglects to identify how this mediation occurs. Research has identified the use of Ontologies to support mediation [CXH04], and when used in conjunction with DAML-S supports resource mediation. When mediation is negotiated between services there is now a need to represent new processes using metadata. OntoServ should have identified available workflow languages to represent the agreed integration. However, OntoServ neglected to identify that DAML-S Process Model provides the necessary metadata for automatic validation of resources in the architecture according to [APS04]. Using BPEL in this architecture would have provided a reliable metadata to support process registration and execution.

2.2 Semantic Web enabled Web Service Project

The Semantic Web enabled Web Service project (SWWS) [BFM02] aims to transform the web from an information source into a distributed service resource allowing services to easily interact with the support of machine processable and interpretable metadata. SWWS exploits the evolution of semantic web languages to support automatic service discovery, selection and execution in a business context, thus enabling the automatic composition of complex processes using appropriate service descriptions. From their E-commerce perspective all web services must be able to trade and interact with other commerce outlets, so that using their methodology will provide a flexible approach for such interaction. To allow interaction between E-commerce entities, numerous heterogeneous and semantic issues must be resolved and this motivates the focus of their work. To utilise their metadata service, SWWS provides semantic based architecture which complements their Web Service Modeling Framework (WSMF) [FB02a, FB02b]. To offer full flexibility of industrial commerce their metadata service and conceptual architecture is centred on two key prin-

ciples: firstly, maximum de-coupling of components in an E-commerce application and secondly, strong mediation to enable all entities interact.

To enable the SWWS principles, there are a number of requirements particularly in relation to process composition that must be addressed by the technologies that are now introduced:

- Any new process must be modelled and easily executed.
- Transmissions between service partners in a process require security according to their B2B requirements.
- All relevant services must be easily discovered and interacted with.
- Different document types must be mediated to allow for integration.
- There is a need for process flexibility to support adaption, to meet the overall goal.

With respect to these requirements, the relevant technologies applied by SWWS will be discussed. As document types act as business documents such as purchase orders in the service context, there is a requirement to describe structure and semantics to allow partner mediation and interaction. SWWS propose the use of XML for document exchange described by the Ontologies, adding knowledge based metadata to document types to support automation through machine interpretable Ontologies. Semantic descriptions are not only applicable to document types, but are also required to support discovery, interaction and integration to add machine understanding to these components. To address B2B security concerns messages are exchanged using encryption. The Web Service Modelling Framework and Conceptual Architecture will now be discussed.

2.2.1 Web Service Modeling Framework

The Web Service Modeling Framework (WSMF) is a framework for describing web services to enable full and easy E-commerce based on the previously discussed SWWS de-coupling and mediation principles. To create a web service enabled E-commerce platform using P2P technology, there are number of obstacles that WSMF must address. Firstly, there is a need for easy discovery and validation of vendor service offerings, which requires

the support of Semantic Web based technologies to enable automatic vendor discovery. However when a vendor is discovered, heterogeneous message formats must be integrated by using mappings between message schemas. Secondly, in the composition of complex business processes using existing web services, mediation is required to compose this new process and represent the integration necessary for service co-operation. The WSMF consists of four components: Ontologies, Goal Repositories, Web Service Descriptions and Mediations which will now be discussed to address these E-commerce issues.

Ontologies interweave human understanding into a machine processable format. They are formal and conceptual formalisations that can provide the necessary meaning to domain and specific objects to support automatic machine understanding. In respect to WSMF, Ontologies allows the definition of formal WSMF terminology semantics to describe different elements of the WSMF specification. This Ontology based terminology is reused throughout the architecture to provide the necessary semantics for the E-commerce platform.

WSMF describes a goal as the "objectives that a client may have in case he consults a web service" [FB02a]. Client goals are described under their precondition and post-condition. The precondition describes what the web service requires to provide the service, while the post-condition describes what the service returns from an input. WSMF separates the goal specifications from the service descriptions using many-to-many mappings between them as many web services can serve the same goal. Ontologies are used to support a goal specification by providing the necessary semantics to create mappings between services and the goal repository.

WSMF considers the inherent complexity of describing decomposed process working used by DAML-S as trivial as it does not distinguish between internal and external service descriptions and instead adopt their own black box approach to create their own description for web services. Their web service description is composed of twelve elements detailed in Table 2.2 to replace existing web service metadata technologies WSDL and DAML-S.

Property	**Describes**
Name	Unique identified for the service
Goal Reference	The purpose the web service fulfills.
Pre-condition and Post-condition	Conditions for execution of web service.
Input and Output Data	Structure of input and output data.
Error Data	Indicates problems or error states.
Interaction Data	Invoking of other services to provide the goal.
Data Flow	Dataflow between interacting service is a service.
Execution sequence	Execution and control flow for the execution of services.
Exception handling	Returns from service fails.
Compensation	Options available when service fails.
Message Exchange Protocol	Message protocol for service inputs and output.
Non Functional Properties	Other web service properties.

Table 2.2: WSMF Web Service Description Properties

Mediation is a key component in supporting adapters, thus providing flexibility in the SWWS E-commerce platform, allowing web services communicate with heterogeneous sources. WSMF identifies four key areas where Ontology based mediation can support adapters to overcome heterogeneity: data structures, business logics, message exchange protocols and service invocations. As data structures provide message schemas there is a requirement to adapt service outputs to service inputs within the process. While business logics contain many inherent complexities in relation to interaction with other services, mediation will maintain the required integrity. However as some message exchange protocols are not applicable to all E-commerce applications, WSMF uses mediation to allow services communication irrespective of protocol. To allow automatic process composition mediation enables dynamic service invocation to provide full E-commerce flexibility. For an open E-commerce environment, this level of mediation is required to cope with diverse heterogeneity.

2.2.2 SWWS Conceptual Architecture

The goal of the SWWS Conceptual Architecture is to complement the WSMF to provide a methodology allowing automatic discovery, invocation and composition of services for E-commerce. This architecture illustrated in figure 2.2 taken from [FB02a], adopts a 3 layered approach to enable semantic enabled web service applications composed of:

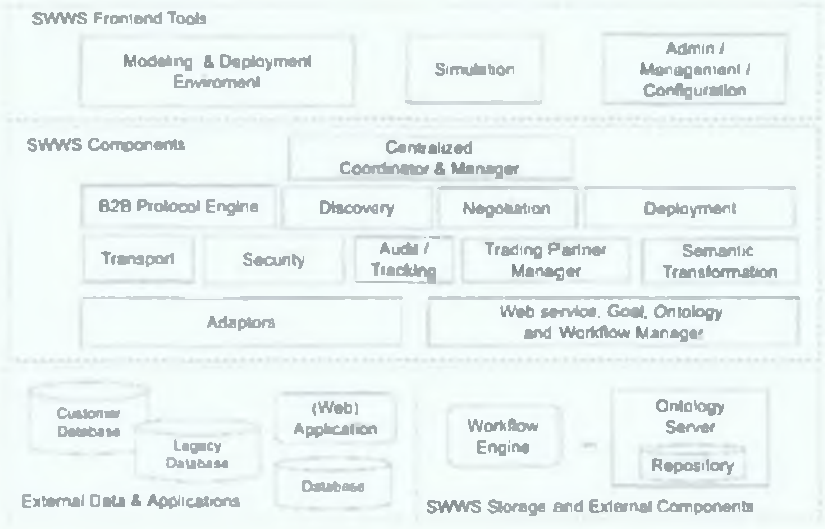


Figure 2.2: SWWS Conceptual Architecture

- SWWS front-end to support administration, management and deployment of services.
- SWWS components providing the required functionality for Semantic Enabled Web Services.
- SWWS storage provides storage for web service descriptions.

The SWWS Components layer is the core entity providing the necessary functionality for E-commerce under this architecture. This layer contains all the relevant functionality to enable Ontologies discover, invoke and compose new services, while maintaining the necessary security for E-commerce cooperation. To support efficient discovery SWWS proposes using distributed UDDI registry over P2P technology. Composition is achieved through the Centralised Coordinator and Manager who controls all components in the layer, enabling new processes to be easily composed using discovery, negotiation, semantic transformation, deployment and audit elements. Invoking of services is supported by the deployment element creating a web service view for composed processes. To allow for easy composition and interaction between services, mediation is imperative. Consequently, SWWS enhances mediation in this layer using adapters with the support of semantic transformation and the web service manager.

2.2.3 Limitations

Despite the benefits proposed by this architecture to provide automatic discovery, invocation and composition of services, the architecture and model are built upon an unspecified Ontology, with no proven success in real world application. This model neglects to take advantage of the Ontology Web Language for Services (OWL-S) successfully implemented to support similar aims in the following projects [SPA+03, PSS+04, BCP05] . Instead SWWS opted to specify its own Ontology for services with little proof of its benefits over OWL-S. Although their work aims to use current technologies, they neglect the most fundamental description available for web services i.e. WSDL. This language provides users with a common interface for accessing and invoking services detailing inputs, outputs, binding and communication information. In relation to process composition they describe the need for a model to represent a composed process, but provide no concrete representation for this scenario, instead offering to include this description in their undefined web service Ontology. The main limitations of this work is their lack of standards particularly in relation to WSDL, defining their own specifications with little implementation detailing how their works offer greater semantics to Semantic Web Enabled Web Services. WSDL is a well known and understood language by web service users and developers and to replace this language with a complex Ontology could become unmanageable in a large E-commerce environment.

The SWWS architecture offers potential in providing an E-commerce layer for semantic web services, specifying the necessary user interaction at the front end and offering efficient storage of metadata at the back end. The components described in the middle tier could provide automatic discovery, invocation and composition required by SWWS. However by providing an unspecified Ontology for describing services and inadequately describing the components functionality in relation to their model, requires further implementation and proofs to add integrity to their aims.

2.3 The METEOR-S Project

Acknowledging the increasing popularity of Web Services to address organisations B2B and EAI concerns, the METEOR-S[VSS+04, POS+04, SVS+03] project adopted a semantic web service architecture and framework to support service discovery, composition and orchestration. Concerned with the lack of industry acceptable standards to semantically describe web services enable automation, METEOR-S first focused on enhancing WSDL, adding semantic web representations to support automation using this description. Consequently, as web services had been described using both WSDL and DAML-S, METEOR-S developed a Web Service Annotation Framework to semi-automatically annotate WSDL descriptions with Ontologies for service discovery. However to support composition, semantic WSDL descriptions were utilised inside the METEOR-S Web Service Composition Framework (MWSCF), to automatically compose and register new processes from existing services. To maintain scalability and autonomy for distributed web services METEOR-S created a scalable P2P infrastructure for service registries. These contributions will now be addressed in respect of the requirements of XLIM.

2.3.1 Adding Semantics to WSDL

Due to the complexity of semantic web service technologies and the reluctance of industry to adopt these technologies, METEOR-S decided to semantically enhance currently accepted the industry standards WSDL and UDDI to automate discovery. To semantically enrich WSDL constructs, DAML+OIL [HvHPS+01] ontologies were added. These extensions map service operations described in WSDL to concepts in appropriate DAML+OIL ontologies. Message parameters which are inputs and outputs were identified for ontology descriptions to allow machine interpretable representations of their XML Schema descriptions. Annotating ontologies to these descriptions provided more expressiveness allowing parameter concepts to be easily shared and understood. When composing processes from existing services there is a requirement to understand existing service preconditions and effects to compose valid processes. As WSDL does not provide for these semantics, METEOR-S extended WSDL to add precondition and effect children to WSDL operations,

with these children referencing ontology descriptions.

To enable discovery of these semantically annotated descriptions, METEOR-S provides a UDDI query interface to handle the semantic annotation. In order to represent semantic WSDL descriptions in UDDI, METEOR-S used `tModels` and `keyedReferenceGroup` UDDI components to group the necessary semantic representations. This representation enabled semantic information to be represented in four `tModels`. The first `tModel` represents ontology descriptions of the functionality, with the second and third `tModels` represent input and output ontologies respectively, while the fourth `tModel` represents the grouping of operations to inputs and outputs. By semantically enriching the UDDI interface, this supported the automatic discovery of available services.

2.3.2 Web Service Annotation Framework

To limit user effort in adapting semantic web technologies for web services, METEOR-S proposed a semi-automatic approach in their Web Service Annotation Framework [POS+04] to create annotations between WSDL and ontology descriptions. The framework described a three layered approach to enable semi-automatic annotation composed of Ontology-Store, Translator Library and Matcher Library layers. The Ontology-Store stores all ontologies to be annotated by WSDL descriptions. These ontologies are categorised into domains creating subsets of ontologies for analysis. The Translator Library adopts the conversion tools provided by METEOR-S to convert WSDL and Ontology descriptions to a METEOR-S `SchemaGraph` enabling mappings to be presented over the graph representations. The `SchemaGraph` representation of a chosen WSDL file and corresponding domain ontology descriptions are then fed into the Matcher Library. The Matching Library supports both element level and schema matching algorithms, which are supported by user interaction. The result of this process is an annotated WSDL document which can be registered with the METEOR-S UDDI.

2.3.3 METEOR-S Web Service Composition Framework (MWSCF)

METEOR-S proposes the use of Semantic Process Templates to semantically represent the requirements of composed processes. These templates describe processes offering activity,

control flow, conditions and calculation descriptions to be interpreted by a process execution engine. These templates provide the process semantics required by the METEOR-S Web Service Composition Framework (MWSCF). This framework is composed of four components: the discovery infrastructure, process builder, XML repositories and process generator.

METEOR-S Web Service Discovery Infrastructure (MWSDI). As all web services are advertised in registries, potentially thousands of distributed P2P registries are available. METEOR-S adopts a specialised ontology called the Registries Ontology which contains the relationships between all domains in the discovery infrastructure and the associated registries. The Registries Ontology improves discovery techniques enabling service requesters query only relevant domains.

Process Builder. The MWSCF specifies a semi automatic mechanism for process composition. The first stage in this process requires the user to compose a generic semantic process template specifying the activities and control flows of the new process. The semantic process template constructed by MWSCF using both BPEL and MWSCF, combines to generate a semantically rich executable process. In addition, a WSDL representation of the new process is automatically created. To transform a generic process template into an executable process, MWSCF adopts a service selection and ranking mechanism. This requires the discovery of applicable services using the enhanced UDDI, while the semantic WSDL descriptions described in §2.3.1 will be processed to calculate the overall semantic matching value. Selected services will then be ranked based on semantic matching analysing inputs, outputs, preconditions and effects and QoS criteria. Once all the matching results are presented, the process creator links data and control flows for selected services to the process template and generates an executable process represented in BPEL.

XML Repositories. MWSCF manages a pool of XML repositories to support the storage, searching, sharing and reusing of ontologies, semantic process templates and service interfaces described in WSDL. MWSCF adopted a native XML database approach for storing these repositories.

Execution Engine. The generated BPEL representation is registered with a BPEL4WS engine. This provides a powerful mechanism enabling BPEL representations to be transformed into invocable processes.

2.3.4 METEOR-S P2P Infrastructure for Registries

METEOR-S adopts a P2P infrastructure for UDDI registries to provide a reliable architecture for METEOR-S Web Service Discovery Infrastructure (MWSDI) components while offering scalability and reliability advantages. To support the MWSDI four peers are specified: the Operator Peer, the Gateway Peer, the Auxiliary Peer and the Client Peer. The Operator Peer maintains a UDDI registry, provides services for the registry and provides Registries Ontologies for peers. Gateway Peers act as an entry point for Peers joining MWSDI, with the responsibility for updating the Registry Ontology and propagates Registry Ontology updates to all peers. To enhance access to the Registry Ontology, Auxiliary Peers act as providers. To support user interaction with MWSDI, Client Peers are allowed to instantiate MWSDI components. This infrastructure provides a reliable architecture for clients and programs discovering services supporting semantic based registry discovery.

2.3.5 Limitations

This project offered a number of advantages for Service Oriented Architectures, providing the necessary semantics to WSDL to support semantic discovery through a customised UDDI interface. The Web Service Annotation Framework provided a mechanism to easily create semantic WSDL descriptions. The METEOR-S Web Service Composition Framework allowed for the easy composition of new processes adopting BPEL and native XML databases. Both of these technologies are core components in the XLIM project, supporting discovery and composition. The adoption of a Peer-to-Peer architecture is advantageous for discovery improving scalability, reliability and illuminating single point failures.

Although this project adopted the semantic web concept to annotate WSDL, METEOR-S eliminated DAML-S too early due to lack of implementation and did not review its developments over the scope of the project. Subsequently DAML-S has become OWL-S, and

is now under the control of the W3C. The OWL-S framework has been adopted by many research Service Oriented Architectures to support the automatic discovery of services and process composition, unlike the semi-automatic approach adopted by METEOR-S. The role of OWL-S in Service Oriented Architecture has expanded to support automatic process validation according to [APS04]. Research described in [TP04] detailed how an OWL-S representation can be transformed to a BPEL executable representation. Furthermore, OWL-S tools have emerged to support the usability and interaction with OWL-S descriptions including [EDM+05, EJ04]. Although the storage of ontologies and WSDL descriptions is achieved by using native XML databases, the METEOR-S project does not focus the discussion on storage requirements and querying of these descriptions at database level to support discovery and composition. However, in XLIM we present a storage architecture for service metadata and describes how queries at database level supports the XLIM requirements.

2.4 Web Service Modeling Ontology Project (WSMO)

The primary motivation for this research emerged from the weaknesses identified in the existing Web Service Architecture [BHM+04] which relies on SOAP for message exchange, WSDL for service descriptions and UDDI to advertise services. As these technologies do not explicitly describe the functionality or semantics of a service they cannot depict the meaning of information being exchanged. In order for web services to achieve distributed computations over the Internet supported by dynamic composition, discovery, selection and mediation, the WSMO project [dBBD+05] was proposed addressing these issues through a meta-ontology describing the necessary aspects of a service. This research enhances the existing Web Service Architecture to overcome the existing ambiguous service descriptions, non-standardised messaging formats and to allow for the composition of service agreements between heterogeneous services.

In the context of this research WSMO focuses on describing the four main elements of web services:

Ontologies. Ontologies provide an explicit specification of WSMO service terminology to enable the linking and matching of real world service semantics. Adopting Ontologies for all resource descriptions offers improved understanding for both humans and machines while supporting automation through Ontology mediators adopting reasoning techniques. In order to capture the necessary resource properties the WSMO provides the following Ontology elements: `hasNonFunctionalProperties`, `importsOntology`, `usesMediator`, `hasConcept`, `hasRelation`, `hasFunction`, `hasInstance` and `hasAxiom`.

Web Service Description. Web Service Descriptions semantically describe the functional, non functional and behavioural aspects of a service. A `webService` class is adopted by WSMO to describe services containing `hasNonFunctionalProperties`, `importsOntology`, `usesMediator`, `hasCapability` and `hasInterface` elements to enable service requesters interpret the service functionality and to support the resolution of integration problems through a mediator.

Goals. A goal represents the objective sought by the service requester. Service requesters provide a goal class describing their desired functionality and behaviour semantically. This model supports the matching and mediation of service between goals and descriptions. The elements provided in the goal class are `hasNonFunctionalProperties`, `importsOntologies`, `usesMediator`, `requestsCapability` and `requestsInterface`.

Mediators. Mediators are adopted by WSMO to solve interoperability problems occurring between class elements. WSMO defines four different mediators for connection WSMO elements: ooMediators to connect and mediate interoperable ontologies, ggMediators to mediate between goals analysing the source and target goals, wgMediators mediates between goals and web service descriptions and wwMediators mediates between two web service descriptions. All mediators are described in the class mediator containing the following elements: `hasNonFunctionalProperties`, `importsOntology`, `hasSource`, `hasTarget` and `hasMediationService`.

2.4.1 Limitations

WSMO offers a new and interesting approach to semantically describe web services, thus supporting automatic discovery of services through mediators. Unlike the OWL-S approach which relies on three sub-ontologies profile, process and grounding, WSMO separates the elements needed to describe services into Ontologies, Web services, Goals and Mediators. WSMO provides a clearer distinction when representing choreography and orchestration. However this research adopts the current standard BPEL for describing web service based orchestration, overcoming any limitation posed by OWL-S. WSMO offers greater support for mediation by explicitly defining mediators in the model. Although WSMO offers many advantages when describing web services, this technology lacks the maturity to represent choreography and grounding adequately. Contrary to WSMO, this research focuses on adopting current best practice to offer a solution to enable automatic discovery, execution and composition of services. The lack of adequate grounding specification in WSMO to incorporate WSDL at the stage requires further research before this language can offer a valid solution to support automatic discovery, execution and composition of services in an architecture.

2.5 Conclusions

In this chapter some leading projects on resource integration were examined in order to identify their metadata technologies and roles. During this study the advantages of using certain metadata technologies approaches became apparent, although some limitations were observed, particularly where suitable technologies were not exploited. In order to provide an efficient metadata service for Service Oriented Architectures an analysis of these projects potential to meet our requirements is now discussed:

1. Heterogeneous Resources. To overcome independent distributed heterogeneous resources there is a need to provide an intelligent metadata, containing all the semantic information to enable accurate understanding of data and resources to compose new processes. Therefore the metadata service should deploy an OWL-S repository

detailed by OntoServ to describe web services in conjunction with supporting Ontologies classes discussed by SWWS to facilitate automation and promote machine interpretable data.

2. Independent Entities. As institutions, countries and schools are independent entities in respect to the Bologna Agreement, maintaining a need to manage and coordinate their own resources requires a scalable and robust architecture. OntoServ has identified Peer-to-Peer networks as a model for creating this large scale distributed independence managed for the architectural purposes by a central mechanism, the OntoShell.

3. Discovery. SWWS and OntoServ identify the need for intelligent discovery using Ontologies to allow process composers easily find and understand required services. However contrary to the SWWS unspecified Ontology, OWL-S profile model has been identified my many researchers [SPA+03, PSN+03] as providing the necessary semantics for accurate resource discovery. Another important aspect of discovery is registry location. OntoServ adopts a P2P approach to this problem provide discovery mechanisms on centralised controllers OntoShells. To optimise discovery of a large finite quantity of resources, P2P structures will be investigated to support the Bologna Agreement.

4. Process representation and execution. A rich composition metadata language is crucial to describe generic processes, overcoming integration and protocols issues. The role of BPEL to provide executable descriptions was discussed by the METEOR-S project identifying the ability to deploy an executable process for its metadata description. This approach offers a powerful tool in a distributed environment, enabling existing services to be discovered and orchestrated into a new process which can be automatically deployed.

The research projects presented in this chapter have all attempted to offer Semantic Web based solutions in conjunction with industry standards to promote automation in Service Oriented Architectures. Each project presented their relevant technologies and detailed their importance to support automatic discovery, execution and composition for services.

These approaches offered significant contributions, but remained unable to support full automation in Service Oriented Architectures by not utilising relevant technologies. Contrary to these limitations, this research has adopted the necessary technologies to enable automation in Service Oriented Architectures, selecting mature and proven technologies. Another focus neglected by these projects was the storage of chosen metadata technologies for processing. This research addresses this issue by providing applications with direct access to relevant metadata, thus reducing the need for parsing and searching large quantities of irrelevant XML data.

At this point the advantages of existing web service technologies have been discussed, providing a platform to develop a metadata service for processes and services in the Bologna Declaration. It was concluded from these projects that although they all offered a significant contribution for their niche problems, there requires further efforts in identifying and investigating the potential of other metadata technologies. Therefore our work will focus on combining existing metadata technologies to gain the maximum potential for discovery and process composition. We will also seek to provide a comprehensive metamodel to cover all metadata aspects of service descriptions.

Chapter 3

Service Architecture

In the previous chapter, research projects presenting Service Oriented Architectures were analysed. This analysis focused on their architectural models and the benefits offered by these metadata models to support service discovery and process composition. The main constraints of these projects was the inadequate use of appropriate metadata technologies, to offer increased and required functionality. To overcome these issues it is necessary to define a new architecture with accompanying metadata which will be discussed in this chapter.

Web Service technologies are designed to allow integration providing interoperability between often diverse applications. Current industry metadata such as WSDL and XML Schema offer discovery, description and communication details enabling interoperability. However these standards alone do not provide the flexible composition of business processes. BPEL, the workflow definition language composes business processes representing the integration between these diverse web services.

In parallel to the development of industrial technologies, the Semantic Web [MSZ01] community has developed metadata languages to recognise the growing need of the richer specification of semantics for web services, more flexible automation and the creation of tools and methodologies allowing web services to be easily discovered, composed and invoked. Efforts supporting this included the creation of metadata technologies such as RDF [GK04], RDFS [BG04] and more recently OWL-S an Ontology language for web services. Unfortunately much of the earlier work on the Semantic Web ignored industrial

standards and created opposing architectures.

The primary focus of this chapter is to present the distributed architecture of XLIM and identify the metadata technologies required to support adequate and accurate discovery and integration of services.

Outline. This chapter is organised as follows: the XLIM Architecture is presented in §3.1; while the required metadata technologies to support the XLIM Architecture are introduced in §3.2. The Metadata Management Framework is detailed in §3.3 to display the management and functionality support by XLIM. The chapter will end with conclusions provided in §3.4.

3.1 XLIM Architecture

The inadequacies of the client/server architecture are apparent, when requiring large scale resource availability, performance and integration. The Peer-to-Peer model provides an alternative by distributing the main costs through sharing data, but maintains a centralised registry for querying purposes. The limitations of a centralised UDDI registry becomes clear in a large scale distributed environment due to scalability, consistency and ambiguous semantics. The argument for Peer-to-Peer based registries has received much research focus due to its inherent advantage of sharing resources and query optimisation identified by [PKY03], while unambiguous semantics are required to realise the automation on the Semantic Web. Exploiting the advantages of distributed registries for large scale resource integration, we present a Super Peer registry model to support service mediation that overcomes diverse heterogeneous resources while facilitating quick response, composition and execution times. Additionally, rich semantics are provided to support discovery, automation and growth.

Consider the problem scenario introduced in §*1.3* created by the Bologna Declaration. Before entering third level education, the student has chosen to study in Dublin City University, University of Montpellier and University of Esbjerg. This presents serious technical and pragmatic issues for the institutions and their countries requiring the deployment of

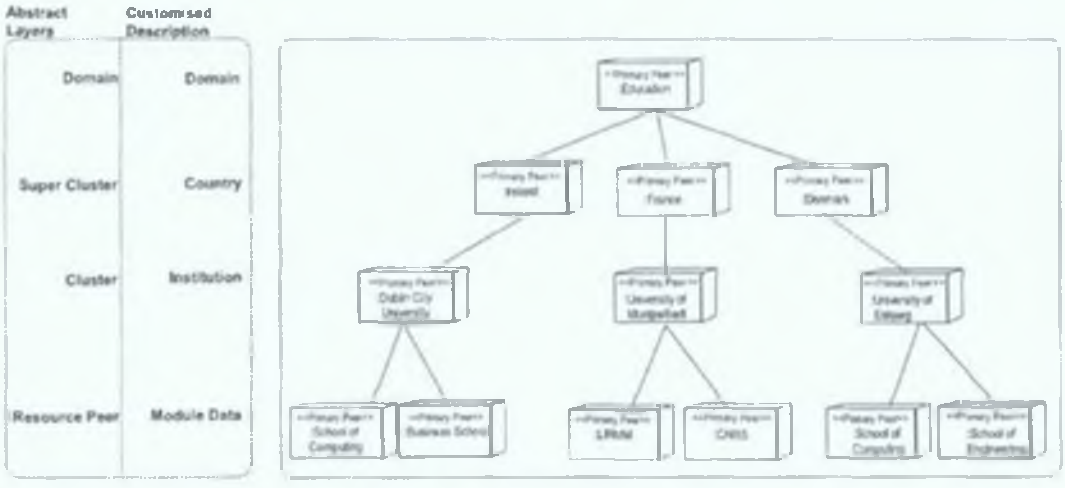


Figure 3.1: Bologna Super Peer Deployment Diagram

a robust architecture to meet the distributed and semantic challenges. The approach in XPeer[MR04, RB04] was to construct clusters of services and course modules using different requirements. For example, all computing networking modules in Ireland, all Java subjects delivered in French, Computing degrees in France not requiring Mathematics, and Wireless computing topics in Europe.

Like all Super Peer networks, peers in XLIM contain metadata describing other peers in their domain. However, XLIM adopts Ontology descriptions to represent peer functionality in a specific domain. Using Ontologies for content based addressing detailed in [CFM+03], enables XLIM to automatically compose super peer instances enabling discovery and composition. XLIM adopts a system of abstract layers to meet these requirements, where a scenario or group of scenarios will create an instance of these abstract layers, as in *figure 3.1*.

An instance view of the architecture can be viewed in *figure 3.1*, outlining the super peer classifications in XLIM. All XLIM peers that provide resources are considered reliable and must advertise their resources in a semantic registry to enable automatic discovery and composition. Consequently, this architecture supports distributed registries containing taxonomies of services inside domains discussed in [VSS+04]. The paradigm of Peer to Peer networks to support registries offers a number of benefits. Since each registry is an independent entity, it allows for greater autonomy, enabling each domain manage its own

services. The independence of Peer-to-Peer registries also allows for easy setup and maintenance in the architecture. The super peer model was employed to represent the domains and sub domains required for the clustering of similar data to support timely results from the database. The XLIM project provides an extension to the work on XPeer by clustering services and metadata inside the domains represented in XPeer, while maintaining the role of XML in both data and metadata flows.

3.1.1 Super Peer Clustering

The XLIM architecture provides abstract super peer classifications `domain`, `super-cluster`, `cluster` and `resourcepeer` to represent a large scale distributed resource architecture. This classification and clustering of registries over a super peer network is a natural extension from the Peer to Peer concept to enhance the access, control and maintenance of each peer. Classification allows for improved structuring of information storage space, discovering of services and pruning the search space. To deploy an Education instance for the student wishing to study in Dublin, Montpellier and Esbjerg as described in §1.3, XLIM adopts the customised layer descriptions of `domain`, `country`, `institution` and `school`, to provide services applicable only to the appropriate layer and peer while supporting quick access the registry. This clustering approach allows nodes easily join and leave the architecture without effecting existing nodes. Nodes joining the network must register with the layer directly above and provide access to their registry of services for querying and duplication purposes. By supporting the duplication of node registries nodes can freely leave without affecting the architecture as duplicate registries are deployed to replace the removed node. The classification and decentralised approach makes the XLIM architecture scalable as the number of registries increase. These levels of clustering which will now be discussed in respect to the Bologna example.

Domain Layer. The *domain* layer in the architecture acts as controller for the designated domain and in the Bologna Declaration context, this is Education. This layer has a number of roles to support large scale service querying, composition and execution. To meet the minimum and future requirements of a European Education system, the domain

layer specifies education rules and services required to join this domain. An sample service required by the Education Peer would be a query service allowing users query the entire domain. These services are composed from the layers below with the support of workflow and semantic metadata. The registry contains services applicable to this level enabling more efficient querying based on levels. Querying is further enhanced by offering a domain level addressing service for the registries, providing queries with access to their required registry.

Super Cluster and Cluster Layers. The role of the *country layer* and *institution layer* is to provide comprehensive clustering enabling more efficient resource querying and usage. Like the domain layer, a semantic registry is maintained detailing all country and institution services including supporting services required by the domain or upper layers. These layers maintain control over the layer directly below and joining peers must implement required services and adhere to the rules specified by the layer above. Depending on the instance scenario different levels of clustering are required to provide lower peers with required rule, service and Ontology classes. A country resource rule could be France requiring all students to have gained 10 credits in French subjects before they can attend university in France. In this scenario two levels of clustering are required with France acting as the Super cluster providing rule data to the institution on the cluster layer, while the resource layer contains Module data.

Resource Peer Layer. The *module data layer* is required to provide access to all module data resources. However this layer must implement the mandatory XLIM services and rules outlined by its cluster layer peer.

This classification, clustering and control of peers in the XLIM super peer architecture offers the scalability and performance requirements to successfully manage a large scale resource architecture dispersed over many heterogeneous sources.

3.2 Reuse of Current Standards

There are many metadata technologies used by research and industry to support web services in large architectures and there is much discussion as to their suitability. XLIM requires intelligent metadata to allow services advertise themselves to support querying, composition and access. The aim of this section is to introduce the technologies that underpin the XLIM architecture. For the remainder of this thesis all resources described in XLIM are represented by the XLIM *metadata model* containing WSDL, OWL-S and BPEL descriptions.

3.2.1 Web Service Description Language (WSDL)

The WSDL [CGM+] language presents a standard XML format providing information about the interface and semantics of how to invoke web services. WSDL provides requesters with four critical requirements to invoke services: interface information describing available functions, input and output messages represented by XML Schema data types, binding information detailing communication protocol and addressing information for locating the service. A WSDL specification acts as a contract between service providers and requesters providing necessary service data offering platform and language independence. A web service description using WSDL is modelled in two parts: an abstract definition of the service includes `message`, `operation` and `interface` data while concrete definition details `binding`, `service` and `endpoint` data. In summary, WSDL provides requesters and providers with a highly flexible and robust language to describe a web service.

3.2.2 Ontology Web Language for Services (OWL-S)

To support the Semantic Web paradigm there is a need to include greater semantics in the construction of more powerful tools and methodologies enabling more flexibility and automation of services. OWL-S (formerly DAML-S) developed by DAML [MBH+04], specifies an OWL based ontology for web services, resulting in a rich representation description that provided a comprehensive specification of many aspects of a service. This provides semantics offering better support to automatic service selection, invocation, composition

and transmission of messages between heterogeneous services. As a web service has a Service Profile, Process Model and Grounding, OWL-S provides the necessary semantics to describes these features.

Service Profile. The OWL-S framework provides the Service Profile to describe the service offered by the providers, supplying the necessary information for service requesters. The machine processable ontology class Profile describes the function of a service consisting of three types of information: the organisation that provides the services, the service function and the host of features that specify characteristics of the service according to [MBD+03]. The main role identified for the Service Profile is supporting a discovery service, allowing requesters query resources based on their required profile.

Process Model. The Process Model provides a detailed view of how the service operates or specifically, operations with users and other services. The key aspect of the Process Model is its representation of Inputs, Outputs, Preconditions and Effects. Representing these key functions of services in the Process Model supplies requesters with workflow data, allowing them to select the most appropriate provider. The Process Model allows for Atomic, Simple and Composite Processes to be described. Atomic Process are directly invocable describing no subprocesses and are executed in a single step. A Simple Process is not invocable, but is conceived as being single step execution. These can be specialised Atomic Processes or an abstract representation of a Composite Process. A Composite Process is composed of a number of Atomic or Composed Processes and are decomposed through the OWL-S control constructs, detailing process interactions and relationships. This Model is effective in providing the requester with the interaction protocol, detailing the data and actions to support the invocation or composition of web services.

Grounding. Grounding details how the requester can interact with the services, specifying the communication protocol, message formats, serialization, transport and addressing information necessary for service implementation. The Grounding class complements the WSDL language in the framework to take advantage of the complementary strengths of the two specifications.

3.2.3 Business Process Execution Language (BPEL)

BPEL [ACD+03] (also BPEL4WS) provides a rich vocabulary for representing the behaviour of business processes used for composition, orchestration and coordination of web services and to allowing for easier integration among business partners [Jur04]. BPEL allows two distinct processes to be described: an executable and an abstract business process. The executable process provides all details of the process, while the abstract process details only the public message exchange between the parties. Like OWL-S, BPEL also relies heavily on WSDL to support the invocation of web services in a business process. BPEL provides a comprehensive vocabulary allowing for complex algorithms of service workflow supporting synchronous and asynchronous communication described using XML syntax.

BPEL is supported by a BPEL engine [IBM02] allowing for the compilation, execution and deployment of business processes. The engine allows for flexible bottom-up or top-down approaches to process composition, synchronised XML source and tree views of the business process and validation of process against specification requirements during editing. During process registration the BPEL and WSDL documents are consumed to create a single access point for requesters wishing to invoke the BPEL process as a web service.

3.3 Metadata Management Framework

In order to realise the benefits of the metadata discussed in this chapter it is necessary to introduce a Metadata Management Framework (MMF) provided by each peer to manage the interaction of metadata to meet user requirements. The concept of the MMF is to provide: easy registration and storage of metadata, discovery of peer services, easy composition and execution of processes; maintenance and management for processes; and a dynamically growing repository of processes to meet user needs. This methodology consists of four main elements:

- Ontologies

- Queries
- Mediators
- Process Execution

As detailed in the previous section, the OWL-S ontology language is the key enabler in service discovery and composition, by providing the mandatory web service semantics. Ontologies specified using OWL [SWM04] interweave human and machine understanding to provide automation. Ontologies describing the key vocabulary applicable to European education are shared among all peers. Content based addressing is defined through Ontologies as detailed in [CFM+03], to provide content location-aware routing.

Queries define a service problem initiated by the user and a result can either provide a registry match, or could possibly compose a process. In the second case, the query must adequately define the objectives and mappings to be performed in order to match objectives with available services. Tracking will identify regular user queries to dynamically increase the services available to meet user requirements, enabling new processes to be composed and registered.

Mediation is required to overcome the inherent heterogeneous entities in the business environment. The role of mediators is to negotiate between users and peers to compose new processes. Another requirement for mediation is in relation to new peers joining the architecture: they must ensure that a new peer contains relevant resources and meets the requirement of joining the system as specified by the parent peer. To allow older systems to provide resources to peers, mediation will be required through the use of an adapter provided by peer owners. The adapter to create a web service interface over these systems to support the XLIM web based architecture will be described in *chapter 5*.

Execution of newly composed services is required to meet changing requirements. BPEL provides the markup language available to execute processes from OWL-S composition results. OWL-S results are then transformed to BPEL processes using XSLT. Creating new processes with BPEL allows for easy deployment and registration in the XLIM architecture.

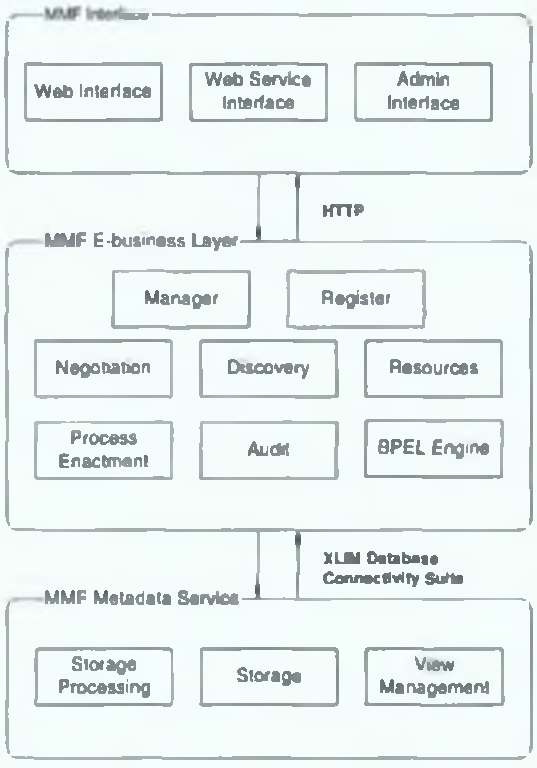


Figure 3.2: Metadata Management Framework

3.3.1 Conceptual Peer Architecture

The conceptual architecture is now presented to provide the necessary support for process composition and maintenance. This architecture has been designed to enable our semantic vision, allowing dynamic growth and automatic maintenance of current processes. The conceptual architecture displayed in *figure 3.2* depicts the Metadata Management Framework available at each primary peer. The architecture is composed of three layers which will now be introduced.

MMF Interface Layer. Provides for human interaction with peers, allowing peer managers and user collaborate with peer resources. Peer managers can configure and administer the e-business layer through a configuration interface, allowing new services to be registered with associate Ontology descriptions and new Ontologies to be specified. A web interface provided for users, facilitates interaction with services or processes available in the service registry. HTTP provides the transport protocol from the interface layer to the

E-business Layer.

MMF E-business Layer. Creates an E-business Layer over existing and new resources to provide user desired services. This layer takes full advantage of the semantic and workflow metadata provided to support automation and is responsible for implementation of the 4 key elements of this architecture. All communication with the MMF Metadata Service Layer is controlled by a collection of Metadata Service classes represented in the XLIM Database Connectivity suite. A full description of this layer is provided in *chapter 5*.

MMF Metadata Service Layer. Proficient storage of MMF metadata is necessary to provide quick access to a metadata repository containing possibly terabytes of service metadata . To achieve this XLIM provides a Metadata Service to store and manage all metadata registered on a peer. The role of this Metadata Service Layer in XLIM will be addressed in *chapter 4* describing storage and view management.

3.4 Conclusion

This chapter has laid the foundations of the Metadata Service, detailing the metadata technologies needed to meet the e-business requirements forced upon academic institutions by the Bologna Declaration. After creating the original super peer network in XPeer to improved load balancing and robustness over centralised hybrid system, the XLIM architecture adopted this approach for resource repositories supported by metadata technologies. These metadata technologies are fully integrated into the architecture to play a fundamental role allowing service discovery and composition, descriptions supporting service invoking and workflows creating a processable view of composed services. The deployment of super peer metadata repositories allowed peers to maintain control over their metadata, while achieving the high level of performance necessary. A detailed view of the control, registry and composition at each level of the architecture, supporting efficient querying based on peer or domain required. This chapter concluded by introducing the Metadata Management Framework presenting the E-business layer for service discovery and integration and the Metadata Service for peer storage of service metadata. The MMF

will now be fully specified in the next two chapters detailing the Metadata Service and the E-business Layer. The storage of service metadata by the Metadata Service will be discussed.

Chapter 4

Metadata Service

The previous chapter provided an overview of the XLIM architecture and introduced the associated web service metadata technologies that act as the core data components for the metadata service. While metadata offers detailed resource semantics and workflow data, the XLIM architecture highlighted the requirement for metadata to support *resource integration*. Due to the large quantity of resources available by all European institutions participating in the Bologna Declaration, it could create terabytes of resource metadata over distributed peers. Maintaining and querying large quantities of data causes many performance issues which must be addressed by the Metadata Service. In this chapter a Metadata Service is presented to manage metadata for querying and access purposes. Hence, the Metadata Service must address storage requirements to facilitate the XLIM project.

Outline. This chapter is organised as follows: the Metadata Service is introduced in §4.1 in conjunction with the storage mechanism; while the metadata storage is presented in §4.2 detailing the storage of service metadata discussed in *chapter 3*. To maintain the integrity of metadata represented by the Metadata Service the metadata views will be described in §4.3 and the conclusions are provided in §4.4.

4.1 Metadata Service Overview

Accurate identification of resources is essential for resource integration to match requirements and integrate resource data. As detailed in *chapter 3*, the metadata model supports the research requirements of accurate discovery, resource integration, process creation and process execution. However, in order to provide users and programs with access to their required information the metadata service will address the storage and access components of metadata management. Due to the XLIM requirement of querying possible terabytes of resource metadata the key challenge is the correct factoring and cataloguing of associated metadata to provide efficient discovery. Thus a reliable, high performance metadata service is required which will be presented here as the XLIM Metadata Service and the metadata storage mechanism.

To allow the successful discovery of resources, XLIM users must be able to easily query resource metadata and retrieve accurate results. However to achieve this, XLIM must implement the following requirements in the metadata service:

Store and share metadata. To allow users discover and interact with resources, the Metadata Service must provide efficient storage of resource metadata to enable efficient querying. Consequently the Metadata Service provides an architecture for metadata storage referred to as *storage models* which will be discussed in §4.2. Certain metadata, such as domain specific should be easily shared in the XLIM architecture to allow all users the necessary access to their required domain data.

Organise metadata in a logical fashion for efficient discovery. As there is potentially terabytes of resource metadata available throughout European institutions, it is necessary to organise data in a logical fashion using architectural and storage models to achieve this. Utilising metadata in storage models will facilitate the querying of large repositories more efficiently.

Enable users create views. View mechanisms play an important role in information systems providing users with a subset of the data repository while maintaining the integrity

and autonomy for the underlying database schema. The metadata service is required to facilitate view management by enabling views to be created that do not effect structure or authorisation integrity but provide view storage for quick access to data subsets.

Manage large quantities of data. Due to the potential quantity of data, the Metadata Service must be able to manage and query large quantities of XML data. To support this an appropriate database and indexing model must be chosen.

4.1.1 Conceptual Metadata Service

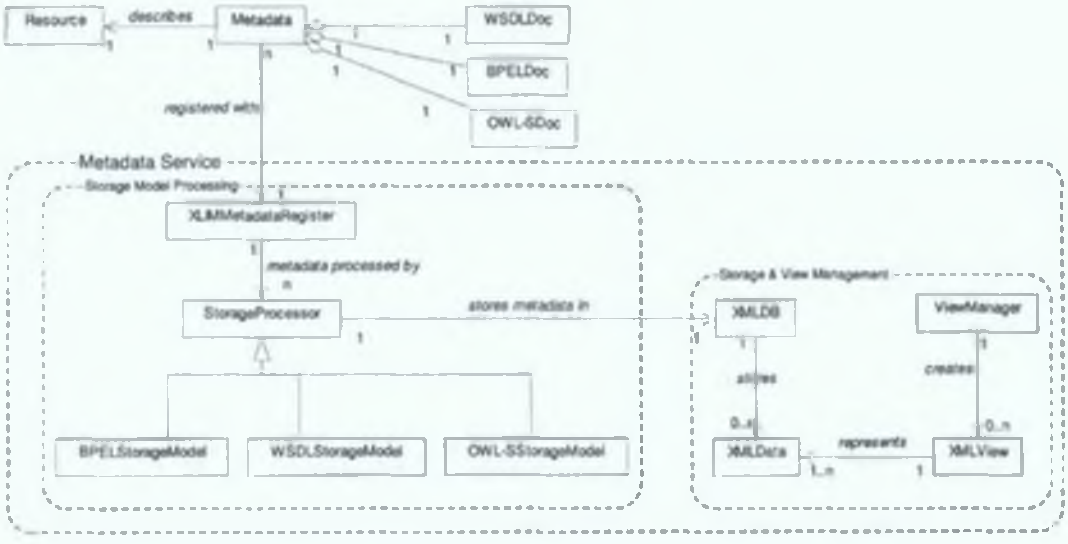


Figure 4.1: XLIM Metadata Service Model

These requirements leave many challenges for the XLIM Metadata Service which must be addressed in the implementation model now presented. In the previous chapter the XLIM architecture was presented detailing the location of institutions and schools in a *Super Peer* network. As each peer is responsible for its own resources, it is imperative there is an effective metadata management solution for data intensive metadata. The XLIM Metadata Service provides storage processing and view management functionality at a peer level for service metadata as described in *figure 4.1*.

As detailed in *chapter 3*, when a service is registered in XLIM, its accompanying metadata model containing OWL-S, WSDL and BPEL metadata must be published with their as-

sociated peer in the XLIM architecture. As all of the metadata types are semi-structured, meaning they conform to a predefined XML Schema, this provides the opportunity for the Metadata Service to exploit this characteristic to offer a storage architecture to increase efficiency. The Metadata Service displayed in *figure 4.1*, demonstrates the XLIM metadata registration process. The XLIM Metadata Service is composed of two components to manage service metadata at the storage level. Firstly, metadata processing segments the inputed metadata into the XLIM storage models. Secondly, processed metadata is stored in a specified XML database allowing view creation to provide users with repository subsets of required data. These components will be presented in the remainder of this chapter.

4.1.2 XML Storage Options

As the numbers of XML documents in this system can grow rapidly increasing the importance for quick access to relevant XML data, a storage mechanism is required to offer increased access and data management functionality. There are many standard storage systems that must be compared to provide the necessary functionality and performance to meet XLIM metadata management requirements. Many options have been researched including relational, object-relational databases and native XML databases. These approaches will now be discussed in respect to [SF03] and[FWF03], who evaluated the functionality and the performance offered by these approaches.

Relational Database Storage. Storing documents in relational databases requires the translation of XML nodes into hierarchical, tree-structured relational schema linked using relations. This approach requires significant processing and time to develop an appropriate schema matching all nodes while maintaining the necessary relations. Furthermore, once the schema is created further processing is then required when adding new documents as these must be parsed and encapsulated into schema. From a querying perspective, relation SQL results in large join overheads as required relationships must be joined to retrieve results.

Object Relational Database Storage. Object relational storage for XML documents requires the development of object relational schemas based on a predefined XML Schema. Once the object relational schema is created all new documents must be parsed and mapped to the new schema thus increasing the processing time for new documents. Unlike the relational approach this approach offers significant performance advantages for storing XML document as join costs are reduced as XML is utilised in object relational schemas. Current object relational storage models for XML have deployed XML-Enabled databases storing XML in object relational formats offering XML querying functionality. This approach offers many benefits for data management including update and rollback functionality, greater security and reliability and allows querying using both SQL, XPath and XQuery. However research conducted in [O'C04] outlines the disadvantages of Enabled XML Databases describing no schema evolution supported, memory management and resource issues and a lack of support for internal DTD subsets. This research concluded by detailing the challenges for deploying complex XML Schema in enabled XML databases, forcing the database to quit during the schema registration stage.

Native XML Database Storage. Native XML databases including eXist [Mei02], Tamino [Sch01] and TIMBER [JAKL+02] offer direct storage of XML data, without any conversion to relational models. This allows for data to be stored and accessed at minimal cost, thus providing direct access to XML data and reducing the processing time of other models according to [FWF03]. The native XML database architecture depicts a three layered approach. At the bottom layer of this architecture the data is stored in its original format. The middle layer offers database management functions through an XML engine supporting the querying and storage of XML data and facilitates the inclusion of current XML technology engines with direct access to the data required. The query language supporting the native XML database projects mentioned above is XQuery, but some projects use a proprietary query language to meet their requirements. The top layer details the interfaces available to the database allowing users access the database through a graphical interface and providing programs HTTP, SOAP and XML-RPC access. These databases are quickly adapting to meet the latest XML and technology needs, quickly adopting new standards into the database system.

4.1.3 XLIM Storage

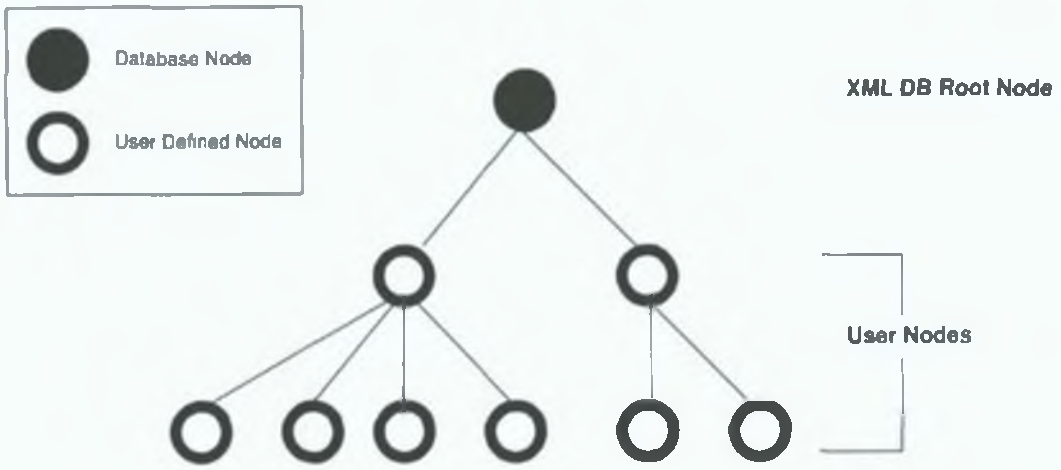


Figure 4.2: Native XML Database Storage Architecture

Native XML databases offer a number of features to enhance XML storage in XLIM which will now be introduced. Collections of XML documents stored in an XML database are managed in hierarchical *collections* comparable to storage on a file system as illustrated in *figure 4.2*. This mechanism provides distinct advantages over a single unified storage location allowing collections to store similar or Schema-based documents, thus enabling queries to target only relevant collections. Languages such as XPath and XQuery have exploited this property, to allow users define queries over documents, collections or sub-collections. The evaluation of structured queries over large document collections of schema-less documents causes a major challenge for query processing. To overcome this challenge and support query optimisation native XML databases use numerical indexing schemes when representing nodes in XML documents, linking nodes and values. This allows for quick identification and analysis of nodes in a large document set.

Although native XML databases lack the maturity of their relational counterparts, native XML databases offer significant advantages for managing and accessing XML data. Native database management systems unlike their relational counterparts have focused on providing management of XML issues, optimising indexing, adopting new XML technologies at database level, enabling the easy integration with XML related tools and providing

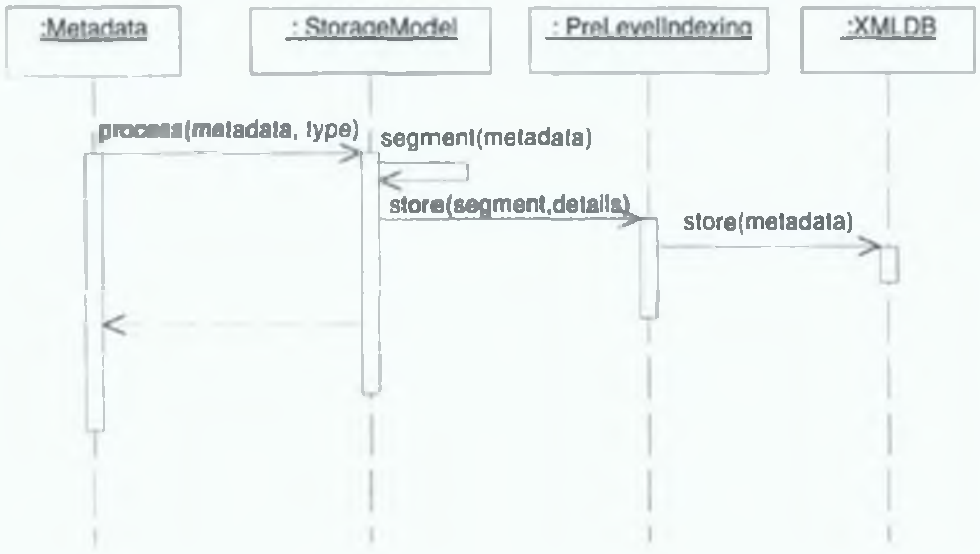


Figure 4.3: Metadata Registration Sequence Diagram

the necessary XML access for Internet and web service components. Research performed by [FWF03] and [SF03] have identified the role and advantages of native XML databases for accessing and updating large quantities of data. As this research is focused on the storage and management of large and distributed quantities of XML data, the next section will identify how the Metadata Service exploits the native XML database storage architecture, to facilitate XLIM resource queries.

4.2 Metadata Processing and Storage

In this section the exploitation of collection structures will be presented to offer increased data access for the XLIM requirements detailed in *chapter 1*. This concept requires manipulating metadata structures to support discovery, negotiation, composition, invocation and auditing, while maintaining the relationships and integrity of the data. The Metadata Service illustrated in *figure 4.3* processes the metadata into segments for storage into relevant collections, including the creation of links between segments . All document registered with the Metadata Service are indexed during storage and this indexing is performed by the PreLevel Indexing Structure detailed in [OBR05, ORB05]. XLIM exploits the col-

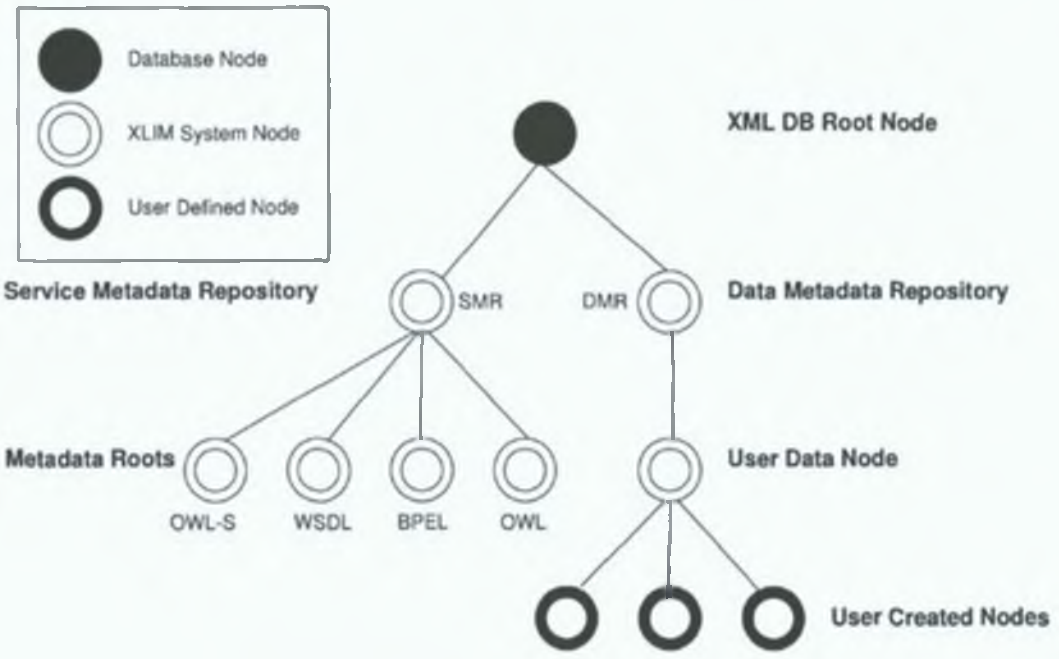


Figure 4.4: XLIM Storage Model Overview

lection feature of native XML databases to provide a hierarchical storage model for XLIM metadata and module data. This extension of the native XML storage architecture increases user access to quantities of similar data for analysis and reduce the requirement to query large quantities of complex metadata documents. As XLIM is focused on providing metadata management for web services supplying OWL-S, WSDL and BPEL metadata, the Metadata Service will now focus on the storage models for these three representations.

Native XML database offers a root Collection as the entry point for the storage of documents in native XML databases. The XLIM storage model extends this model by creating a Service Metadata Repository (SMR) and a Document Metadata Repository (DMR). These models extend earlier work presented in [MR04] to incorporate service metadata into the XLIM storage model as detailed in *figure 4.4* with all service metadata being segmented in the SMR. At the Metadata Roots collections detailed in *figure 4.4*, each metadata technology will be further segmented providing a storage model to enhance access for its chosen requirement.

4.2.1 XLIM Modelling of WSDL Metadata

WSDL acts as the most basic form of web service metadata providing service requesters with the necessary semantics to invoke a service. The properties of WSDL are consumed in whole or part by the BPEL and OWL-S specifications to provide invocation representations of their described services. As a result of practical or whole consumption, the WSDL Storage Model detailed in *figure 4.5* allows BPEL and OWL-S specifications to easily reference the required properties within WSDL document. The WSDL Storage Model segments WSDL documents into the following collections:

Types Collection. This collection contains all XML Schema definitions used by messages in the `message collection`.

Message Collection. All message endpoints required by services are stored in the message collection depicting both input and output messages. Messages stored in this collection either define their own message XML Schema type or reference XML Schema types defined in the `type collection`.

PortType Collection. All operations supported by a service are defined and described in the `PortType collection`. Each operation links to messages defined in the message collection.

Binding Collection. All binding information including transmission protocol detailed for service operations defined in the `PortType collection` is stored in the `binding collection`.

Service Collection. For operations defined in the `PortType collection` and binding specified in the binding collection, the `service collection` stores location details for services providing the URI of a chosen service.

As WSDL is consumed in whole or part by both OWL-S and BPEL the WSDL Storage Model provided allows these metadata technologies to only reference their required WSDL components.

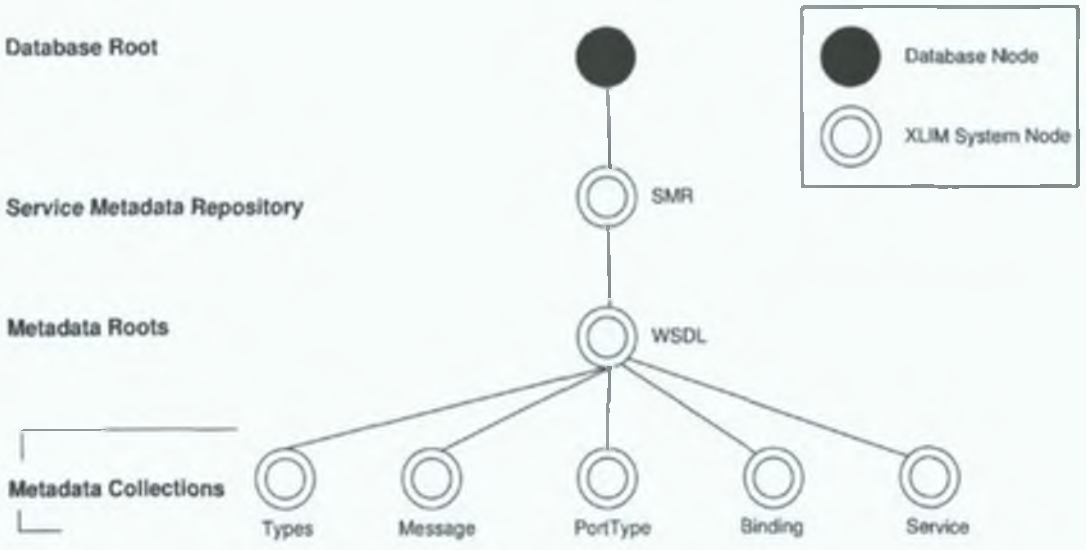


Figure 4.5: WSDL Storage Model

4.2.2 XLIM Modelling of OWL-S Metadata

The XLIM model for OWL-S storage stores all OWL-S metadata in a predefined hierarchical model. The root node of OWL-S metadata is the OWL-S collection detailed in *figure 4.6*. In this collection the upper ontology for services is stored to provide an access point of reference for a service detailing the links to the three OWL-S sub-ontologies described in the collections below. The storage model for each of these sub-ontologies Profile, Process and Grounding will now be described to support the XLIM requirements.

Profile Storage Model

Any service transaction over the XLIM architecture requires the involvement of three parties, the requester, the provider and the supporting infrastructural components. As the service requester is looking to discover services supplied by service providers, there is the need for an intelligent metadata representation for services to allow easy interaction between infrastructural components brokering the service between the provider and requester. The OWL-S Profile sub-ontology provides this necessary interpretable metadata formation for describing services to be discovered, specifying three key service properties: provider information, functional description and service characteristics. To achieve the

rich functional representation for discovery, the XLIM Model for OWL-S Profile segments the Profile properties into collections.

The diagram in *figure 4.6* displays the Storage Model for the OWL-S Profile sub-ontology to support efficient discovery. This is achieved by segmenting the OWL-S Profile Model into seven collections, grouping all instances of a property into the same collection to allow users better query repositories. This model creates repository collections for the following properties:

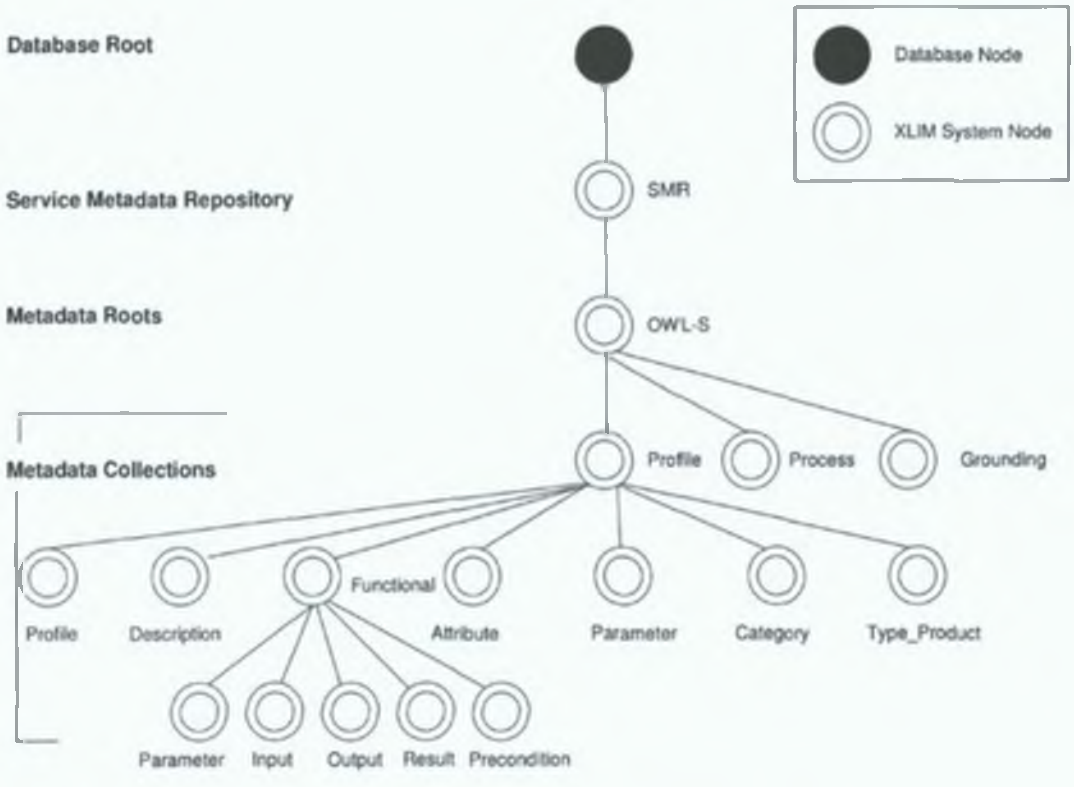


Figure 4.6: OWL-S Profile Storage Model

Profile Collection. The profiles store `presents` and `presentedBy` relationships.

Description Collection. Stores the human interpretable data including `serviceName`, `textDescription` and `contactInformation`.

Functional Collection. The Functional collection is broken down into five sub-collections to represent `hasParameter`, `hasInput`, `hasOutput`, `hasPrecondition` and `hasRes-`

ult data. As these properties are described by the Process Class, the metadata service will ensure the integrity of these constraints are maintained in the database by performing incremental checks.

Attribute Collection. Contains attributes applicable to that service described by the `serviceParameter` and `serviceCategory` properties. These properties are linked to their instances through the data integrity document described in §4.3.

Parameter Collection. All parameters defined in the Attribute Collection have an instantiated representation stored in the Parameter collection. The parameters stored in this collection are described using `serviceParamterName` and `sParameter` properties.

Category Collection. The category of services based on some predefined classification is described in the Category collection. These categories referenced in the Attribute collection are described using the `categoryName`, `taxonomy`, `value` and `code` properties and are stored in the Category collection.

Type_Product Collection. Stores the types and products that are handled by a service. All `serviceClassification` and `serviceProduct` properties are described to provide links to type and product instances stored in the OWL Metadata repository displayed in *figure 4.4*.

This model enables discovery mechanisms to provide requirement based queries on the available repositories through quick access to service metadata, reducing the need for the database to query large quantities of irrelevant data. Once the required or relevant OWL-S Ontology classes are discovered, the entire Ontology class instance can be quickly composed using the data integrity document described in §4.3.

Process Storage Model

As the Process Model provides service requesters with the necessary semantics explaining how to interact with a service, there is a requirement to provide users with quick access to

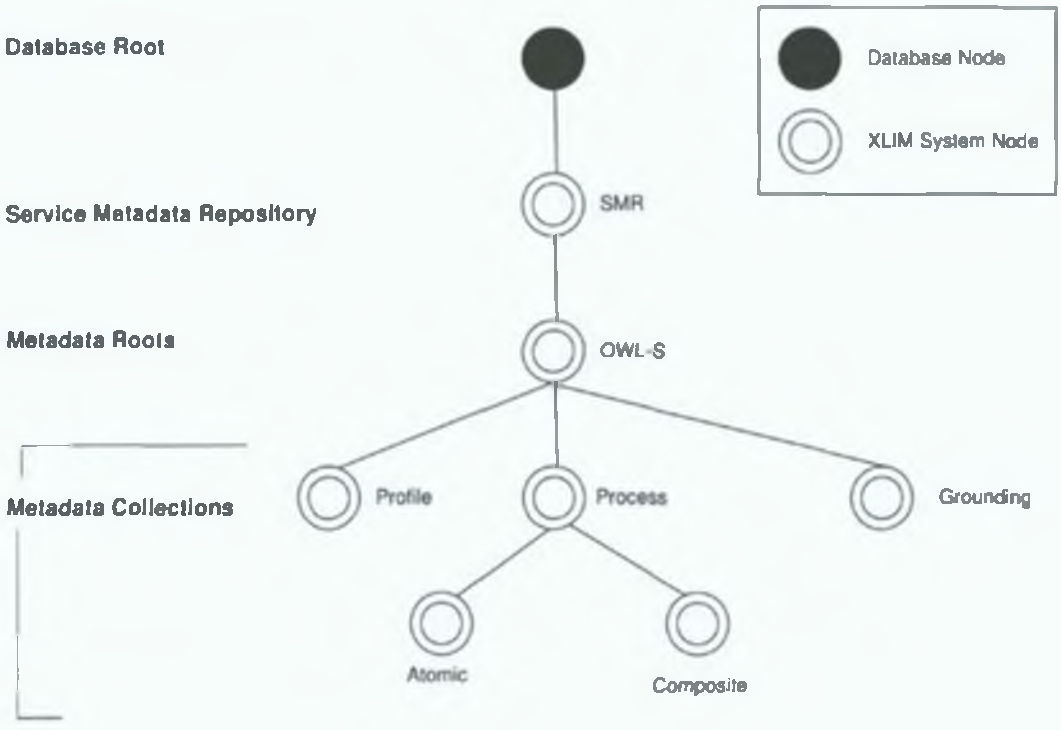


Figure 4.7: OWL-S Process Storage Model

the Process class of their chosen service after discovery. This specification describes service processes under four properties: Inputs, Outputs, Preconditions and Effects to detail how services or processes operate. The OWL-S Process specification allows for the description of atomic or composite of process representations. Atomic and composite representations offer different views of a process and the service provider decides on one or both views to provide as metadata. As a process may contain an atomic and composite view, the storage model uses this criteria for segmentation, which is now described.

Atomic Collection. An atomic instance describes a black box view of a service or process, describing the operation as a single step. Atomic instances present directly invocable services by passing an appropriate message data. In this representation users are not provided with the internal operations of a service or process. Atomic instances of OWL-S Process classes will be stored in the Atomic sub-collection of the Process collection as detailed in *figure 4.7*.

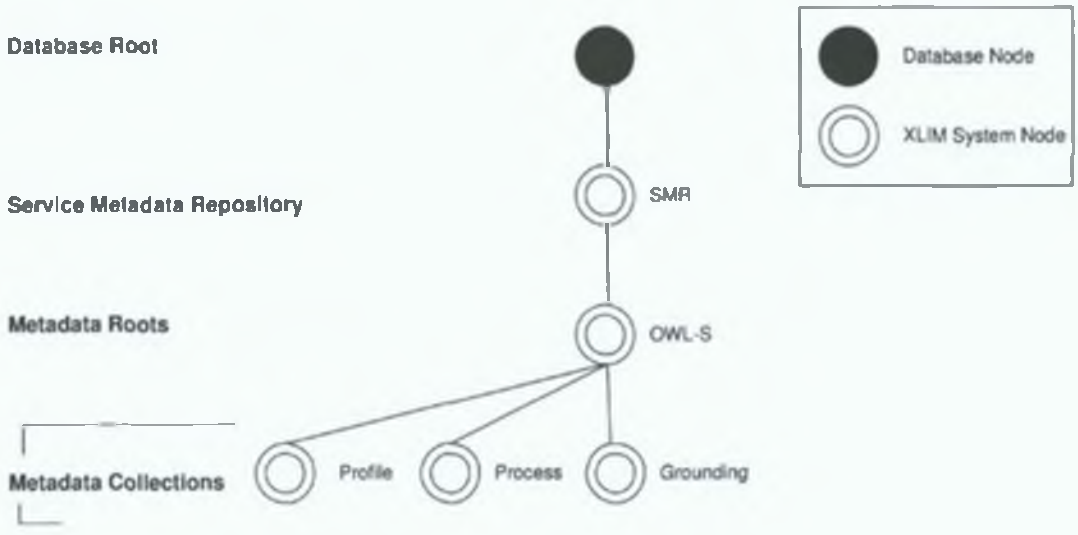


Figure 4.8: OWL-S Grounding Storage Model

Composite Collection. Composite process instances describe a glass box view of a service or process classifying all process operations into their decomposing sub-processes. This representation provides rich semantics for describing processes including sequential, concurrent and condition properties to provide service requesters with a full definition of how the service operates. All composite processes described in OWL-S Process Classes will be stored in the Composite sub-collection as detailed in *figure 4.7*.

The Process model is used by service requesters to describe how the service operates to support negotiation and auditing functionality. Instead of querying large quantities of Process files, these functions only require an atomic or composite process description linked to services identified at the discovery stage. This model offers increased efficiency by providing the service requester with direct access to the atomic or composition process of discovered services, rather than of providing a single file containing multiple representations of unnecessary atomic and composite representations.

Grounding Storage Model

OWL-S Grounding classes are accessed after discovery and negotiation to provide invocation details for the chosen service. As a result, the entire Grounding instance is consumed by the service requester. All Grounding classes are stored in the Grounding collection de-

tailed in *figure 4.8*, to provide service requesters with direct access to Grounding instances for discovered and negotiated services.

4.2.3 BPEL Storage Model

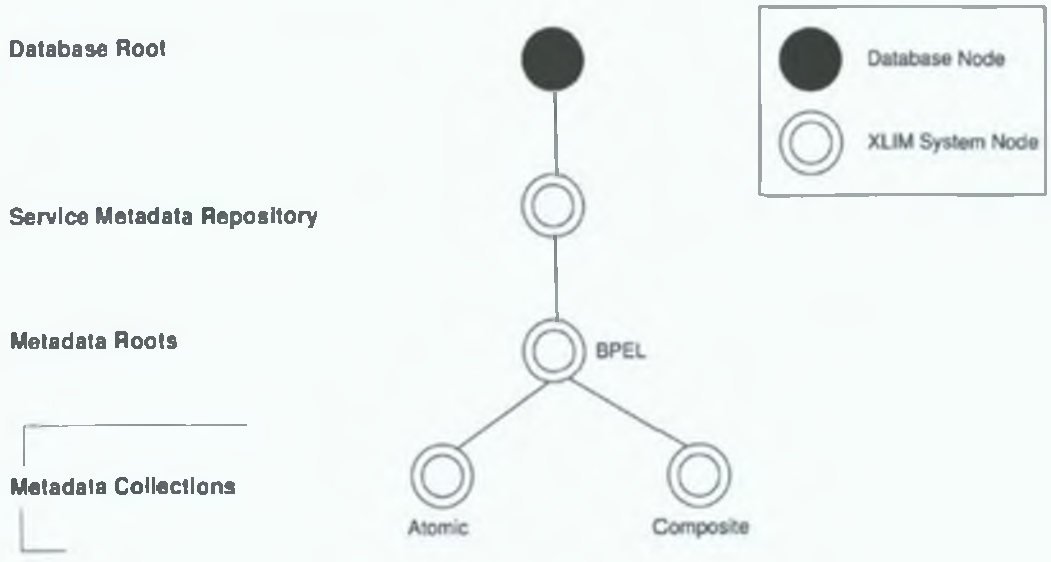


Figure 4.9: BPEL Storage Model

BPEL describes the composition, orchestration and coordination of web services in the XLIM architecture. With similar aims to the OWL-S Process Model, the BPEL specification provides greater workflow semantics for defining services and processes with the added benefit of enabling new invocable processes to be executed through the BPEL engine. As the BPEL engine parses entire BPEL descriptions, any BPEL representation will present this format inside the BPEL Storage Model detailed in *figure 4.9*. The BPEL storage model is composed of two sub-collections described below:

Atomic Collection. All services registered with XLIM must provide a BPEL description to support composition and audit of services and processes. These descriptions detail ultimately one invocable service and are stored in the Atomic collection.

Composite Collection. As all processes composed in XLIM are described using BPEL, an XSLT view representation of the new process is stored in the Composition collection

composing the process view over BPEL representations detailed in the Atomic collection.

The XLIM Model for BPEL segments the two types of BPEL descriptions deployed in the architecture, firstly the BPEL representation of existing services developed by service providers and stored in the Atomic collection and secondly, the system generated process stored in the Composite collection. Creating BPEL process views supports the easy validation of system generated processes in the Composite collection, as BPEL process descriptions can be retrieved from the BPEL engine and validated against the view representation.

4.3 Data Integrity and View Management

It is important for the Metadata Service to maintain the integrity of the documents being consumed into their storage model. To achieve this, XLIM during service registration creates a *service integrity document* for every service or process containing all links and relationships for the service metadata and stores this document in the Service Metadata Repository collection. Consequently the service integrity document creates a one to many relationship for a service to its metadata model segments. This is achieved by adding an `ID` attribute reference to the service integrity document in all metadata segments, while one service integrity document references all of a service's metadata model segments. This document's schema displayed in *figure 4.10*, represents all metadata properties referencing their associated document address inside the XLIM collections. Each node in the service integrity document described in *figure 4.10* contains a text node detailing the address location of that property. This allows the service integrity document to be easily accessed by any query language, program or service requesters wishing to link discovered data to the service metadata model, thus supporting the composition of a service metadata view of a discovered service and providing quick access to BPEL and OWL-S representations.

Figure 4.10: XLIM Service Integrity Document Schema

View mechanisms play an important role in metadata services, providing users with a subset of the data repository while maintaining the integrity and autonomy for the underlying database schema. The requirement of view management in XLIM is necessary to allow users create metadata model views for services over the XLIM Storage Models. The Metadata Service provides a view system to perform this function using XSLT[Kay04] for defining views. XSLT provides a rule based language allowing the creation of complex views from XML documents. Acknowledging the dominance of XSLT for XML view creation online, we investigated the role of XSLT in XML view management in [Kin04].

XSLT provides a view definition language over XML database collections providing users with the functions to transform multiple XML documents into a single XML view. XSLT defines template rules over XML data which have considerable power to transform XML documents and are used here in XML view management. To extract the necessary nodes, template rules adopt the XPath language to provide the XSLT transformation direct access

to the required data. XSLT facilitates the metadata service and developers to create views which meet their specifications. The language contains the necessary semantics allowing for the easy selection, extraction and manipulation of XML data. XSLT transformations act as a stored query for XML views which can be easily accessed in XLIM to provide users with quick access to required data.

The Metadata Service requires views to support service metadata integrity and process composition. The view manager incorporated into the metadata service creates a mechanism for both system and user views to be created over the service metadata storage models. Process views detailing all services and interactions in a composed process are stored in the Composite sub-collection of the BPEL Collection. Whereas all system views are stored in the Service Metadata Repository to provide views over the metadata repositories stored in the collections below. The view manager was necessary to support the successful implementation of the Storage Models. To enable efficient access to a chosen service metadata, a generic service view definition is created in the Service Metadata Repository using XSLT to transform service data integrity documents to full XLIM metadata model representations, thus providing a combined service metadata representation for service requesters and programs.

4.4 Conclusions

In this chapter, we discussed how the XLIM project focused on the problem of storing and organising possibly terabytes of XML based service metadata to provide access to metadata. To address metadata access the XLIM Metadata Service was presented to provide a mechanism for metadata storage and retrieval. However, due to the large quantities of XML based, metadata an applicable storage mechanism was required to ensure quick access to user required data. Native XML databases were chosen due to their advantages for accessing and updating vast quantities of XML data. To facilitate metadata access, the Metadata Service was composed of three components: Metadata Processing, Storage and View Management to organise and manage resource metadata stored in an XML database.

Metadata Processing and Storage provides the necessary functionality to classify regis-

tered metadata into the storages models detailed in §4.2. These models are requirement based, providing data retrieval for their required usage in XLIM, offering increased querying efficiency and access to service metadata. A full storage model view is available in *appendix A*. To maintain integrity over the storage model, XLIM presented a service integrity document. This document maintained the relationships between all segments in a metadata model. Finally the Metadata Service provided a view system taking advantage of the XSLT template based approach to creating XML views. This view system enables both system and users to create metadata views, thus increasing access efficiency by providing users with subsets of required data. The remaining topic for discussion centres on how the prototype was created and how it delivers the specifications outlined in chapters three and four. This prototype forms the basis of the discussion in *chapter 5*.

Chapter 5

Metadata Service Prototype

The Metadata Management Framework (MMF) described in §3.3 outlined a three layer approach to metadata management on peers in the XLIM architecture. In the previous chapter, the bottom layer of this framework was discussed. While the Metadata Service provided users with a comprehensive model for storing service metadata, there is a now a requirement to identify the role of the Metadata Service to support the XLIM requirements. The middle layer in the MMF known as the E-business Layer accesses the metadata provided by the Metadata Service to enable the functional requirements of XLIM. In this chapter the E-business Layer is presented to provide users with the necessary functionality to enable the processes required under the Bologna Declaration. Hence the E-business Layer will address metadata based discovery, composition and verification of services and processes.

Outline. This chapter is organised as follows: in §5.1 the MMF E-business Layer is introduced to provide users with functional access to the XLIM metadata models; in §5.2 the role of OWL-S in supporting automatic discovery is discussed. Once the required services are discovered, §5.3 describes how processes are composed and registered for user consumption. To ensure the validity of created processes, §5.4 addresses how the integrity is maintained, while in §5.6 some conclusions are given.

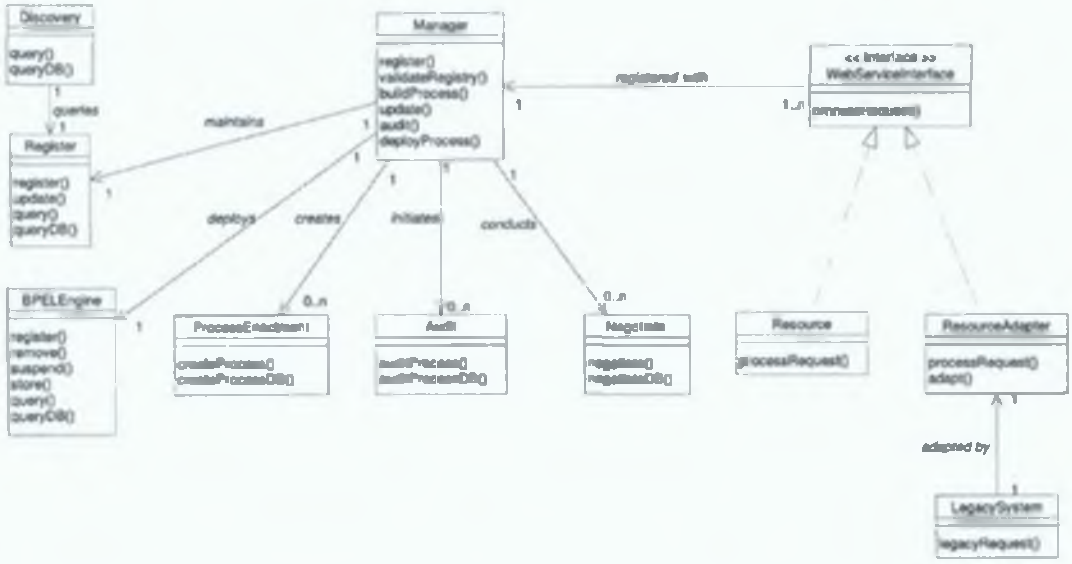


Figure 5.1: e-business Implementation Classes

5.1 MMF E-business Layer

The E-business Layer provides a reliable infrastructure for the adaptation of heterogeneous sources into the architecture enabling efficient querying of peer services, and the supporting and management of process composition and execution. This layer has four functions: it manages metadata in the metadata repositories; it provides the necessary functionality to allow for accurate process composition; it audits existing services and resources to ensure validity; and it provides a BPEL engine for the execution of processes available in the peer. To meet the XLIM requirements this layer offers two approaches to analyse metadata. Firstly, users can adopt a querying approach for analysing service metadata, the E-business Layer supports XQuery queries over the service metadata model. Secondly, users can be provided with their required metadata descriptions to apply existing research on a layer above to interpret and analyse descriptions. A class view of the E-business Layer is described in *figure 5.1* and for the remainder of this section the components of this layer will be described.

Manager. The `Manager` is the central component that incorporates e-business components to allow resource integration in a Peer. All web services in a peer are registered with

the Manager, who stores the accompanying service metadata model using the Metadata Service. To enable process composition the process manager acts as an intermediary with resources providing negotiation and enactment metadata. The manager deploys a BPEL Engine allowing for the registration of new processes into the architecture. The manager is also responsible for ensuring credible services are available to users. To meet this requirement, the manager audits services and processes on an incremental basis. For new peers joining the architecture they must negotiate with the primary peer at the level directly above. The manager of the new peer is responsible for adhering to the requirements outlined in the negotiation.

Discovery and Register. These classes are responsible for registration, publishing and querying of services in the resource peer. The `Register` class provides the `Discovery` interface with direct access to the OWL-S Profile class descriptions provided by the XLIM Metadata Service. To discover services users are provided with two options: `query()` method requiring an XQuery parameter and returns full OWL-S Profile classes for external analysis by [PSS+04] or the `queryDB()` method also taking XQuery as a parameter, but allows users to specify the return values. Consequently, using the `queryDB()` method allows all discovery analysis to be performed at the database level and a specified result such as a WSDL description can be returned.

BPEL Engine. This provides the manager with a deployment tool for instantiating new BPEL processes composed from queries. The `BPELEngine` class will act as a facade, wrapping the functionality of an existing BPEL Process Manager such as [BPE05], to meet XLIM requirements. The BPEL manager acts as a controller for the BPEL descriptions stored in the `BPEL collection` provided by the Metadata Service. This class provides BPEL process management functionality allowing for registration, suspension and removal of processes using the `register()`, `suspend()` and `remove()` methods. While the `query()` method allows users to retrieve BPEL descriptions using location information. However users can query the BPEL directly and customise required results through the `queryDB()` method.

Process Enactment. This creates new processes for the peer created in association with the `Manager` that provides a composed service detailed in OWL-S, which is transformed into a BPEL process and registered with the peer's registry and BPEL engine. The `ProcessEnactment` class provides user access to the `BPEL collection`, `Grounding collection` and `Process collection` provided by the Metadata Service for process composition. The `ProcessEnactment` class supports two types of process composition analysis. Firstly, using the `createProcessDB()` method process composition analysis can be performed at the database level using XQuery. While the `createProcess()` methods retrieves OWL-S and BPEL descriptions for external analysis and composition.

Audit. The `Audit` class ensures the reliability and validity processes in the architecture. The audit mechanism tests service functionality in accordance with OWL-S descriptions available in the repository. Any inconsistencies force the suspension of the process or service. To validate services in XLIM, the `Audit` class must access the `Process collection`, `Grounding collection` and `BPEL collection` for verification purposes, thus this provides an interface to these collections for auditing requirements. The `Audit` class supports two types of verification in XLIM. The first method `auditProcessDB()` is used to support database verification analysing the OWL-S Process class against the process BPEL description using XQuery. While the second method `auditProcess()` retrieves the OWL-S Process class and BPEL description to be analysed externally.

Negotiate. This is used to compose a new process from underlying atomic or composed services. Negotiation is required to provide integration by analysing and matching requirements. Negotiation is achieved using OWL-S Process classes stored in the `Process collection` and Ontology classes stored in the `OWL collection`. This `Negotiate` class supports both database level negotiation with the support of XQuery using the `negotiateDB()` method, while the `negotiate()` returns Ontology classes for external negotiation.

Resource. Resource is a software component supplied by the resource peer to provide for a specific user need. All resources are accessed through a Web Service Interface providing a common interface for all resources in the architecture. A resource can be atomic providing a small specified function or composed containing a group of loosely coupled atomic services, managed by a BPEL definition. These resources are described using the metadata model described in chapter 3 and are stored in a service metadata repository by the Metadata Service.

Resource Adaptor. This class adapts legacy and other older systems into the architecture by presenting these systems as Semantic Web services. Adapters provide users with access to these systems through a web service interface to include the OWL-S ontology and WSDL to support discovery and access. The adapters act as the mediator managing all communication and information exchange with the legacy system. This is based on the generic adapter-component that is configured to wrap concrete software. Similar to most resources the Resource Adaptor must provide metadata model descriptions to be stored by the Metadata Service.

The remainder of the chapter will investigate the role of the Metadata Service to support this E-business Layer.

5.2 Service Discovery

To support discovery requests beyond keyword searches, semantic matching capabilities are used as they also support intelligent automation for service composition. In order to provide discovery and automation in XLIM, semantics must be machine interpretable and queried. Ontology based solutions have become the research standard to support semantic descriptions of Web Services. OWL-S supports a number of activities across the lifetime of the service including greater automation for service composition, service selection, automatic translation between messages in heterogeneous sources and offering support to monitoring and failure. The OWL-S approach supports current web service technologies and enhances them to improve automation.

Semantic interoperability using OWL-S is crucial for discovery in the XLIM architecture allowing web services represent and reason their performed task, explicitly express and reason business relations and rules, understand the semantics of exchanged messages and represent the preconditions and effects of invoking a service. Furthermore OWL-S details wide ranging properties of web services such as quality of service and security in a coherent manner for universal understanding. The OWL-S Profile Model is the primary mechanism for representing this information. In our prototype, discovery is achieved using the `Discovery` class with XQuery, or by retrieving the required Profile class instances for external interpretation. To realise the value added by this class the key properties of OWL-S will be used in respect to the real world example in §1.3. A registration service supplied by Dublin City University allows students to register for semesters. The usage of OWL-S Profile components are now discussed in light of this example before detailing how these component inside XLIM support automatic discovery.

Service Name, Contacts and Description. Accommodates human readable information specifying the service name, owner and a narrative description of the service. The example described in *example 5.1* details service name, contact and description for the Dublin City University registration service is queried to easily determine service owner and name.

Functionality Description. The XLIM Model for OWL-S Profile classes stored in the `Functional collection` provide the functional descriptions of a service specifying the conditions that must be satisfied for a successful result. Furthermore the OWL-S Profile Model specifies the conditions resulting from the service including both expected and unexpected scenarios. To accommodate service access the OWL-S Profile Model represents two functional aspects: the information transformation inputs and outputs, and the state change caused by the service precondition and effect information. The OWL-S Profile Model class defines the following properties for Input, Outputs, Preconditions and Effects: `hasParameter`, `hasInput`, `hasOutput`, `hasPrecondition` and `hasResult`. The underlying benefit offered by the functionality description allows users discover services based on inputs and outputs while detailing the conditions and effects. The Dublin City

```
<profile:serviceName>
   DCU Semester Registration Service
</profile:serviceName>
<profile:textDescription>
   This service allows students register for a semester
   in Dublin City University
</profile:textDescription>
<profile:contactInformation>
   <actor:Actor rdf:ID="DCU-Registry">
     <actor:name>
       Dublin City University
     </actor:name>
     <actor:phone>01 7000000</actor:phone>
     <actor:fax>01 7000001</actor:fax>
     <actor:email>registry@dcu.ie</actor:email>
     <actor:physicalAddress>
        Dublin City University,
        Dublin 9,
        Ireland.
     </actor:physicalAddress>
     <actor:webURL>http://www.dcu.ie</actor:webURL>
   </actor:Actor>
</profile:contactInformation>
```

Example 5.1: OWL-S Profile Class - Service Name, Contacts and Description

University registration service requires a `validateStudent` precondition detailed in *example 5.2* to ensure the user is registered and holds a valid email address. This condition represented in SWRL states that a student must have a relevant email address to use this service, while the rule's properties are semantically described using RDF descriptions to describe `validStudent`, `studentID` and `studentEmail` supporting automatic interpretation and activation.

Profile Attributes. Classification of this service is possible using Profile Attributes providing the users with other aspects of the service. Such aspects can include quality of service, classification and additional parameters to support requester discovery. The example detailed in *example 5.3* specifies a service category for the BOLOGNA-registration. The `profile:code` adds a key relationship to this service, linking to the Bologna Registration Ontology code described in the ontology referenced by the `rdf:ID` to the registration service.

```
<process:hasPrecondition>
  <expr:SWRL-Condition rdf:ID="studentAccountExists">
     <rdfs:label>
          validStudent(studentID, studentEmail)
     </rdfs:label>
     <expr:expressionLanguage rdf:resource="&expr;#SWRL"/>
     <expr:expressionBody rdf:parseType="Literal">
       <swrl:AtomList>
         <rdf:first>
           <swrl:IndividualPropertyAtom>
             <swrl:propertyPredicate rdf:resource="#validStudent"/>
             <swrl:argument1 rdf:resource="#studentID"/>
             <swrl:argument2 rdf:resource="#studentEmail"/>
           </swrl:IndividualPropertyAtom>
         </rdf:first>
         <rdf:rest rdf:resource="&rdf;#nil"/>
        </swrl:AtomList>
     </expr:expressionBody>
   </expr:SWRL-Condition>
</process:hasPrecondition>
```

Example 5.2: OWL-S Profile Class - Functional Description

```
<profile:serviceCategory>
  <addParam:BOLSERV rdf:ID="BOLOGNA-registration">
    <profile:value>Semester Registration Service</profile:value>
    <profile:code>BOL1245</profile:code>
  </addParam:BOLSERV>
</profile:serviceCategory>
```

Example 5.3: OWL-S Profile Class - Profile Attributes

Service Parameter. Provides an expandable list of properties supporting the profile description. The Service Parameter detailed in *example 5.4* details the student parameter linking to an Ontology describing a student in the Irish context.

```
<profile:serviceParameter>
  <profile:serviceParamterName>Student</profile:serviceParamterName>
  <profile:sParameter
   owl:resource="http://www.cao.ie/bologna/owlresources/student.owl"/>
</profile:serviceParameter>
```

Example 5.4: OWL-S Profile Class - Service Parameter

Service Category. Describes a service in relation to a taxonomy of services on the basis of some classification that can be described using a representation such as OWL. Service

```
<discoveredservice>
{
 (for $descdoc in
      collection("/smr/owl-s/Profile/Description/")/Description,
      $catdoc in
      collection("/smr/owl-s/Profile/Category/")/Category/
  let $integdoc := $descdoc/Description/@ID
  where
      $catdoc//BOLSERV/code="BOL1245" and
      $descdoc/Description/@ID = $catdoc/Category/@ID and
      $descdoc//contactInformation/Actor/name/text()
               = "Dublin City University"
  return
  {
      <integritydoc>
      string($integdoc),
      </integritydoc>
  }
  {
      <definitions>
      (doc(doc(concat("/smr/",$integdoc)//wsdl/types)),
      (doc(doc(concat("/smr/",$integdoc)//wsdl/message)),
      (doc(doc(concat("/smr/",$integdoc)//wsdl/portType)),
      (doc(doc(concat("/smr/",$integdoc)//wsdl/binding)),
      (doc(doc(concat("/smr/",$integdoc)//wsdl/service))
      </definitions>
  }
  {
      <processdoc>
      (doc(concat("/smr/",$integdoc)//process/atomic)
      </processdoc>
  })
}
</discoveredservice>
```

Example 5.5: XQuery to retrieve Dublin City University Registration Service

categories are advantageous when searching large distributed repositories as search space will be reduced by referencing the category name. In XLIM there are many categories of services from query services to registration services that are generically described and classified in a taxonomy based scheme.

Service Type and Product. The `serviceClassificaion` and `serviceProduct` specify the type of service and products handled by the service respectively. The values of these properties are instances described in service and product ontologies. Although these

properties are similar to the Service Category they provide full OWL based descriptions, whereas the Service Category need only provide strings referencing a non OWL based taxonomy. The Service Classification and Product of service will link directly to the XLIM Bologna Ontology.

The structure of XLIM storage modelling for the OWL-S Profile model facilitates the easy subquerying of collections containing only relevant data. This distinct feature of XLIM allows discovery queries to span over a smaller set of OWL-S Profile segments to return an XLIM metadata representation of the services discovered. The scenario of the student wishing to study in the universities of Dublin City, Montpellier and Esbjerg (described in §1.3) requires module selection and registration services from each of these universities, with the Bologna ontology class ID for registration services ('BOL1245') already determined. The discovery mechanism must find the relevant registration service for these three universities. The sample query presented in *example 5.5* exploits the XLIM storage mechanism to return the service WSDL description, OWL-S Process location and Service Integrity Document location. The `Discovery` class supports XQuery queries to directly access stored service metadata. However, instead of querying a large repository of irrelevant service metadata, XLIM simply requires the querying of a Description and Category collection to discover all Dublin City University services. This step can be replicated for all universities in the research scenario. Furthermore, XLIM supports more complex querying over its storage modelling based on the Profile components described in this section to achieve further refinement of these results.

5.3 Process Composition

Once the required services to compose a process are obtained, negotiation is necessary to ensure accurate integration. Negotiation requires two key components: Ontology descriptions for data flows and results, and rules to understand preconditions. The OWL-S Process Model classes stored in the `Process collection` describe Input, Output, Precondition and Effect components for registered services. All inputs, outputs and results represented in OWL-S are Ontology based and are automatically interpretable, while rules

are described in OWL-S using the SWRL[HPSB+04] to provide interpretable preconditions for service invocation. Again, using the example in §1.3 requires the analysis of service inputs to achieve the necessary integration between services in a process. The query described in *example 5.5* investigates whether the `StudentName` input required by Dublin City University illustrated in *figure 5.6* and University of Montpellier described in *figure 5.7* correspond to the same representation. To validate this representation, XQuery analyses the XML Schema reference and Bologna Ontology class reference to determine if these representations are the same. However, if these references do not match, the XQuery returns their Ontology classes for analysis.

```
<process:AtomicProcess rdf:ID="DCUSemesterRegistration">
 <process:hasInput>
  <process:Input rdf:ID="StudentName">
   <process:parameterType
    rdf:datatype=
    "&xsd;http://www.bolognadec.org/xsd/studentname.xsd">
     &THIS;http://bolognadec.org/student.owl#studentname
   </process:parameterType>
  </process:Input>
 </process:hasInput>
</process:AtomicProcess>
```

Example 5.6: Input Parameter Described for Dublin City University Registration Service

```
<process:AtomicProcess
   rdf:ID="MontpellierRegistrationService">
 <process:hasInput>
  <process:Input
   rdf:ID="StudentName">
   <process:parameterType
    rdf:datatype=
    "&xsd;http://www.bolognadec.org/xsd/studentname.xsd">
    &THIS;http://bolognadec.org/student.owl#studentname
   </process:parameterType>
  </process:Input>
 </process:hasInput>
</process:AtomicProcess>
```

Example 5.7: Input Parameter Described for Monpellier Registration Service

The integration results achieved in the negotiation stage must be represented in a composite process format for registration and execution. Research conducted in [SCZ04] utilised the OWL-S Process Model classes with associated SWRL rules to facilitate composition

```
<inputresult>
{
 (for $dcudoc in
       doc("/smr/owl-s/process/atomic/dcuregProcess.owl")
             //AtomicProcess[ID="DCUSemesterRegistration"],
     $montdoc in
       doc("http://montpellier.edu/bologna/smr/owl-s/process
           /atomic/dcuregProcess.owl")
            //AtomicProcess[ID="MontpellierRegistrationService"]
 where
     $dcudoc//input/@ID = $montdoc//input/@ID
 return
     if ($dcudoc/parameterType/@datatype =
           $montdoc/parameterType/@datatype) and
         ($dcudoc/parameterType/text() =
           $montdoc/parameterType/text())
     then
     {
       <result value="true"/>
     }
     else
     {
       <result value="false">
       {
         doc($dcudoc/parameterType/text()),
         doc($montdoc/parameterType/text())
       }
       </result>
     })
}
</inputresult>
```

Example 5.8: XQuery To test Similar Service Inputs

using reasoning techniques. The result of negotiation will be an OWL-S Process Model representation of a new process which requires registration into the architecture. However, the XLIM approach can provide these negotiation techniques with direct access to SWRL, OWL-S Process and BPEL representations for discovered services. It is important to note that as a rule service for the XLIM remains outside the scope of this research as all service rules may not be available from existing rules described in the OWL-S Process Model.

As a result, the `ProcessEnactment` class provides direct access to the `Process collection`, `Grounding collection` and `BPEL collection` to support the composition of a composed process representation. Although the OWL-S Process Model semanti-

cally describes the flows between services in a process, the model cannot be executed as the service or process behaviour cannot be predicted prior to execution. Consequently, there is a need to represent services in an executable format for the E-business Layer. The BPEL metadata language introduced in §3.2.3 allows processes to be registered with a BPEL engine [IBM02] and invoked by users. As the result of process negotiation is an OWL-S Process instance accompanying OWL-S Grounding instance, there is a requirement to transform this result into an executable BPEL process which can be invoked by users. Relationships between the OWL-S class and the BPEL representations are analysed and results composed at database level or alternatively these representations can be retrieved to support the automatic transformation to BPEL using a state transition system in conjunction with a Model Based Planner as described in [TP04].

5.4 Process Verification

Verification of composed processes is necessary to ensure correct implementation of processes and to maintain the trust of XLIM users. The verification process audits the properties of a process functional description to ensure the advertised process is valid. As the OWL-S Process Model is organised to describe the interaction between services in a composed process, [APS04] identified the possibilities of validating processes by using this metadata description with an accompanying validation algorithm. To include this functionality, the `Audit` class supports the easy retrieval of OWL-S Process classes. As services in a process relate to each other through data and control flow, the OWL-S Process Model supports a wide range of components which are now described.

Data Flow. Data flows represent the relationships between inputs and outputs of services. The OWL-S Process model describes the data transformations between services using the `hasInput`, `hasOutput` and `hasLocal` properties. These properties describe the necessary mappings between services in the process. The validation of these properties is achieved through Ontology analysis to ensure the semantic meaning has not been altered and by validating outputs against previous successful process invocations.

```
<checkschema>
{
 (for $wsdldoc in
          doc("http://www.dcu.ie/registry/reg.wsdl"),
      $procdoc in
          doc("smr/owl-s/Process/Atomic/dcuregprocess.owl)
               //AtomicProcess
  return
  {
     if(string($wsdldoc//element[@ID="StudentName"] ) =
         string(doc(string(($procdoc)
         \\Input[@ID="StudentName"]\parameterType\text())))
     then
     {
          <result value="true"/>
     }
     else
     {
          <result value="false"/>
     })
}
</checkschema>
```

Example 5.9: XQuery To Test XML Schema representations are equal

Control Flow. Control flows describe the temporal relationship between services in a process. This requires a wide range of components including service sequence, conditions and synchronisation between concurrent processes. These components are described under the `hasPrecondition` and `hasResult` properties and are supported by `Sequence`, `If-Then-Else`, `Choice`, `Repeat-While`, `Repeat-Until` and `Split` sub-properties described in the OWL-S Process Model.

XLIM supports the validation of OWL-S Process classes against associated WSDL and BPEL descriptions at database level. Consider the example discussed in §5.3 verifying the representation of the student name input in Dublin City University and University of Montpellier. The database level supports the verification of the Student Name XML Schema described in the OWL-S Process class against the WSDL XML Schema representation for the Service Input. This verification is illustrated in *example 5.9* which returns a true result if these XML Schema representations are equal, otherwise a false result is returned, specifying the description or process is Invalid.

5.5 Working Prototype

The working prototype for this research built the XLIM Model for service metadata on top of the eXist native XML database. Although this model could have been executed in an Object relational XML Database like Oracle, the conclusions realised in [O'C04], detailing poor memory and resource issues, in conjunction with the expensive validation process highlighted the need for an approach offering optimised performance while reducing validation overheads for large volumes of XML storage. Research described in [FWF03] concluded that native XML databases provide higher performance for handling large volumes of XML data. As a result the native XML database approach was adopted. The prototype utilises the advantages offered by the native XML database to supply an XLIM Interface for services offering both local and distributed access to metadata. The local interface offers XLIM Model creation, document management, validation, service registration and querying functionality to users. While the distributed interface provides a web service for distributed clients allowing view creation, validation, service registration and direct querying of desired service in accordance with the Model.

5.6 Conclusions

This chapter outlined the role of the E-business Layer within the MMF and described how the metadata provided by the Metadata Service can be easily accessed to support the XLIM requirements. The E-business Layer was presented detailing its components and interactions. This layer provides the necessary functionality to support user requirements for services and processes in the Bologna Declaration. To further justify the role of the Metadata Service to support user requirements, the metadata model was matched with discovery, composition and auditing requirements. The advantages of OWL-S Profile Model discovery over keyword implementations was presented to achieve increased discovery accuracy from greater semantics describing service functionality and to support automation. While the XLIM Metadata Service enables the querying of Profile components eliminating the requirement of querying whole Profile descriptions. By adopting the OWL-S Process Model within the Metadata Service this supports the automatic negotiation of discovered

services to compose a new process. However as the OWL-S Process Model is unable to create an executable process, the transformation of OWL-S Process Model to BPEL was presented. Furthermore by adopting the OWL-S Process Model, this will support the auditing and verification of registered processes. XLIM Metadata Service provides direct access to service and process OWL-S Process and BPEL descriptions to support composition and verification. The E-business Layer in conjunction with the Metadata Service provides an efficient operational model for easy service discovery, process composition and maintenance of existing processes. This finalises our effort to provide a Metadata Service and infrastructure to enable interaction and management of services in a large Service Oriented Architecture. The final step is to address areas for future research.

Chapter 6

Conclusions

The aim of this research was to demonstrate that an enterprise architecture and accompanying Metadata Service can support automatic resource discovery and integration overcoming heterogeneous resources on a large scale. Unlike other research projects, this work focused on providing the metadata requirements to support the requirements of discovery, negotiation, composition, registration and verification. Thus, the architecture supported the automatic composition of processes through process composition and verification to meet user requirements. A second objective was to specify a comprehensive metadata model in an XML database and to provide users with quick access to their required metadata. Finally, the role of the Metadata Service in supporting the XLIM requirements was achieved through an E-business Layer. In this chapter an overall summary is presented in §6.1 before areas for future research are offered in §6.2.

6.1 Thesis Summary

In chapter one, an introduction to enterprise architecture was presented detailing their advantages to support organisational practices. However, as a result of over complex architectures and heterogeneous resources, organisations were faced with large implementation and maintenance costs. Service Oriented Architectures emerged to overcome these problems by providing loosely coupled services that could easily interact with heterogeneous resources. Web Services has emerged as the latest service technology, incorporating

XML standards to allow for easier integration between an organisation's processes and to overcome heterogeneous resources. The motivation arose from a large enterprise architecture scenario requiring the discovery and integration of resources on a European scale. Consequently, this research hypothesis focused on the storage modelling of resource metadata to achieve the XLIM requirements for distributed services and processes.

Chapter two investigated several projects analysing distributed Service Oriented Architectures. Each projects' metadata and architecture were analysed to identify their advantages and limitations. Peer networks were adopted by these projects to overcome the scalable, performance and single point failure disadvantages associated with client server networks. They identified the need for service discovery and composition but adopted different approaches. The OntoServ project adopted a purely DAML-S approach neglecting to take advantage of the executable properties of BPEL descriptions. The Semantic Web enabled Web Service project proposed replacing the DAML-S approach with their own Ontology. However, due to the lack of specification and results, the success of this Ontology remains unknown. BPEL was also neglected in this project removing process execution capabilities. METEOR-S adopted an semantically enriched WSDL specification with BPEL to support discovery and composition. Consequently it was decided that a combined metadata approached was required to realise the XLIM requirements.

The Service architecture for the XLIM project was presented in chapter three. The XLIM architecture adopted a Super Peer system of abstract layers to meet user requirements, where a scenario value creates an instance of these abstract layers. These layers were domain, super cluster, cluster and resource, and were used to support service discovery and composition. The clustering of layers was justified to support scalability, overcome failures and provide control mechanisms over the peers below. The metadata model for XLIM services was introduced containing WSDL, OWL-S and BPEL metadata descriptions. Each of these metadata languages were introduced to justify their inclusion in the architecture. To realise the full potential of the metadata model, the Metadata Management Framework was presented detailing a three layer architecture for metadata management on primary peers. The top layer contained user and web service interfaces for access to the Metadata Management Framework; the middle layer offered an E-business Layer; while the bottom

layer supported metadata storage through the Metadata Service.

The Metadata Service of the Metadata Management Framework was presented in chapter four to provide a comprehensive storage model for service metadata. As the Metadata Service supports native XML databases, their concepts were introduced with the advantages offered over traditional RDMS. Storage Models were then presented for each of the metadata technologies represented in the metadata model. The WSDL Model described the storage model for WSDL to match its requirements in XLIM. The OWL-S Storage Model was segmented into its three models: Profile, Process and Grounding, each requiring a specific storage representation. The OWL-S Profile Model is the primary metadata used by the discovery mechanisms, querying potentially terabytes of metadata. The model was segmented into collections for each of its properties, allowing users to query repositories of properties as opposed to entire documents. The OWL-S Process and Grounding were stored in specified collection to be consumed after discovery. The BPEL Storage Model also maintained full representation to support composition of executable processes. In order to maintain integrity of the metadata model, a service integrity document containing all relationships within the metadata model was stored in the Service Metadata Repository collection. Finally, view management was enabled to allow system and users create metadata views, thus increasing access efficiency by providing users with subsets of required data.

The deployment of XLIM metadata was presented in chapter five, detailing the E-Business Layer for the Metadata Management Framework. The E-business Layer provided a reliable infrastructure for the adaptation of heterogeneous resources into the architecture. The E-business Layer detailed two approaches to interacting with stored metadata to enable service discovery, negotiation, composition, registration and verification of services. Firstly, service requesters can provide XQuery queries to determine results at database level or secondly, the E-business layer provides support for existing research by enabling these research projects act on a layer above the E-business layer providing direct database access to metadata instances supporting their requirements. The XLIM Model for OWL-S offers significant advantages over keyword service descriptions including greater semantic understanding of service functionality and the support for automatic discovery. Further-

more the XLIM Model for OWL-S Process classes supported negotiation and verification requirements in XLIM. Negotiation was achieved by focusing on the Input, Output, Precondition and Effect properties of the OWL-S Process Model enabling service requesters easily access this metadata. The role of the verification process to analyse composed processes was investigated, requiring the validation of data and control flows in the OWL-S Process Model. Metadata analysis to support verification was achieved at database level, while external verification algorithms were supported by providing direct access to OWL-S Process instances. Although the OWL-S Framework provides the necessary semantics to support automation, automatic registration of a newly composed process was required for users to access. This registration was addressed adopting BPEL, and transforming OWL-S Process results to a BPEL representation which could then be registered with the BPEL engine to allow users invoke composed processes. This chapter presented the data access infrastructure offered by XLIM to support the functionality required by service requesters.

Full automatic discovery, execution and composition of services remains an unresolved issue in Web Services, however it is important to note that based on current research standards this research presented a metadata model to support a solution. Although this research cannot guarantee automation in all cases, it provides a storage and querying model for the latest service technologies aimed at overcoming this issue.

6.2 Future Research

Service Oriented Architectures are rapidly becoming a standard for enterprise architecture offering organisations loose coupling and easier integration. The web service paradigm has offered a new level of interoperability and integration adopting XML technologies. However, automation within these architectures has to-date remained the focus of the academic sector, due primarily to the complexity of semantic web technologies, lack of proven techniques and lack of performance. These issues are currently being addressed by research academics and the W3C to enable both semantic web and industry technologies co-operate instead of presenting parallel research. The shift to co-operation has enabled the XLIM architecture adopt both semantic web and industry standards to offer increased

functionality. With the evolution of new industry and semantic web technologies this trend will gradually lead to reduced metadata complexities, increased performance and automatic service oriented architectures being adopted by industry.

The XLIM project has addressed the metadata requirements to enable the processes required by the Bologna Declaration. However this function alone cannot provide all of the necessary components to allow services be easily discovered, composed and executed. Areas for future research have been identified during this work which will now be discussed to provide a more complete solution for automatic discovery and composition.

Rule Service. A topic which was only introduced in this thesis, to enable automatic composition of processes rules must be supplied to ensure accurate composition. Although automatic composition is possible using the service rules provided by the metadata service, valid automatic composition cannot be guaranteed in every situation. To overcome this a rule service is required containing domain and process specific rules to support process composition.

Query Service. Large XML metadata repositories requires a query service to exploit the properties and storage models offered by the Metadata Service. The areas that must be addressed by the query service will now be introduced:

- **Query Language.** The adoption of current query languages XPath, XQuery, and OWL-QL must be extended to exploit the storage models provided by the Metadata Service.
- **Query Routing and Optimisation.** This area has been the focus of significant research in recent years with the evolution of Peer networks. Semantic web solutions have been successfully implemented to provide optimised query execution plans and automatic routing in large architectures.
- **Indexing.** An index structure is imperative in this large architecture to enhance performance by providing information to the query processor. The PreLevel indexing structure created as part of the XLIM project should be implemented to support the storage models provided by the Metadata Service [ORB05].

Integration Algorithms. The E-business Layer detailed the role of XLIM metadata to support discovery, negotiation, composition and verification requirements. The adoption of ontology based metadata with industry standards supported automatic discovery and execution. Further research is required to investigate the performance and accuracy of reasoning algorithms exploiting the Metadata Service. The success and performance of these algorithms is orthogonal to the adoption of automatic architectures by industry.

Process Verification. The E-business Layer introduced a database approach to verifying process metadata. However database verification alone can not provide accurate process verification. Further research requires a dual approach to accurately verify processes using both metadata verification, possibly at database level and an algorithmic approach analysing process workflow at execution.

Bibliography

[ACD+03] Andrews, T., Curbera, F., Dholkia, H. et al., *Business Process Execution Language for Web Services Version 1.1*, IBM, 2003, URL http://www.ibm.com/developerworks/library/ws-bpel.

[ACP+01a] Alexaki, S., Christophides, V., Plexousakis, D. et al., The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases., in *SemWeb*, 2001, URL 139.91.183.30:9090/RDF/publications/semweb2001.pdf.

[ACP+01b] Alexaki, S., Christophides, V., Plexousakis, D. et al., On Storing Voluminous RDF Descriptions: The Case of Web Portal Catalogs., in *WebDB*, 2001.

[APS04] Ankolekar, A., Paolucci, M. and Sycara, K., Spinning the OWL-S Process Model: Toward the Verification of the OWL-S Process Models, in *The Third International Semantic Web Conference (ISWC 2004)*, 2004.

[BCE+02] Bellwood, T., Clement, L., Ehnebuske, D. et al., *UDDI Version 3.0*, OASIS, Version 3 edn., 2002, URL http://uddi.org/pubs/uddi-v3.00-published-20020719.htm.

[BCP05] Brogi, A., Corfini, S. and Popescu, R., Composition-oriented Service Discovery, in *Workshop on Software Composition at European Joint Conferences on Theory and Practice of Software*, 2005, URL http://www.di.unipi.it/~corfini/paper/SC05.pdf.

[BFM02] Bussler, C., Fensel, D. and Maedche, A., A Conceptual Architecture for Semantic Web Enabled Web Services, in *SIGMOD Rec.*, vol. Vol. 31(4), pp. 24–29, 2002.

[BG04] Brickley, D. and Guha, R., *RDF Vocabulary Description Language 1.0: RDF Schema*, W3C, 1.0 edn., 2004, URL http://www.w3.org/TR/rdf-schema/.

[BHM+04] Booth, D., Hass, H., McCabe, F. et al., *Web Services Architecture*, W3C, 2004, URL http://www.w3.org/TR/ws-arch.

[BPE05] Oracle BPEL Process Manager Release Notes, 2005, URL download.oracle.com/otndocs/products/bpel/bpelrn.pdf.

[BPSM+04] Bray, T., Paoli, J., Sperberg-McQueen, C. et al., *Extensible Markup Language (XML) 1.1*, W3C, 2004, URL http://www.w3.org/TR/2004/REC-xml-20040204/.

[CFM+03] Castano, S., Ferrara, A., Montanelli, S. et al., Ontology-Addressable Contents in P2P Networks, in *Proceedings of the 1st Workshop on Semantics in Peer-to-Peer and Grid Computing at the 12th International World Wide Web Conference*, 2003, URL http://www.isi.edu/~stefan/SemPGRID/proceedings/4.pdf.

[CGM+] Chinnici, R., Gudgin, M., Moreau, J.-J. et al., *Web Service Description Language (WSDL) Version 2.0 Part 1: Core Language*, W3C, URL http://www.w3.org/TR/2004/WD-wsdl20-20040803/.

[CXH04] Cruz, I. F., Xiao, H. and Hsu, F., An Ontology-Based Framework for XML Semantic Integration., in *IDEAS*, pp. 217–226, 2004, URL www.cs.uic.edu/~fhsu/publications/cruz-ideas2004.pdf.

[dBBD+05] de Bruijn, J., Bussler, C., Domingue, J. et al., *Web Service Modeling Ontology (WSMO)*, W3C, 1.1 edn., 2005.

[DRR+03] Delobel, C., Reynaud, C., Rousset, M.-C. et al., Semantic integration in Xyleme: a uniform tree-based approach., in *Data Knowl. Eng.*, vol. 44(3), pp. 267–298, 2003, URL http://www.csd.uch.gr/~hy561/Papers/Xyleme.pdf.

[EDM+05] Elenius, D., Denker, G., Martin, D. et al., *The OWL-S Editor– A Development Tool for Semantic Web Services*, 2005, URL http://owlseditor.semwebcentral.org/documents/paper.pdf.

[EJ04] Engel, L. and Jaeger, M. C., *OWL-S Plugin for Axis*, 2004, URL http://ivs.tu-berlin.de/Projekte/owlsplugin/index.html.

[Eur99] European Commission, The Bologna Declaration, Online Resource http://europa.eu.int/comm/education/policies/educ/bologna/bologna.pdf, 1999.

[FB02a] Fensel, D. and Bussler, C., The Web Service Modeling Framework WSMF, in *Electronic Commerce Research and Applications*, vol. 1(2), pp. 113–137, 2002.

[FB02b] Fensel, D. and Bussler, C., WSMF in a Nutshell, *Technical report*, Vrije Universiteit Amsterdam, 2002, URL informatik.uibk.ac.at/users/c70385/wese/wsmf.iswc.pdf.

[FvHK+00] Fensel, D., van Harmelen, F., Klein, M. et al., On-To-Knowledge: Ontology-based Tools for Knowledge Management, in *Proceedings of eBusiness and eWork Conference*, EMMSEC, 2000.

[FWF03] Fong, J., Wong, H. K. and Fong, A., *Performance Analysis between and XML-Enabled Database and a Native XML Database*, vol. 1, Addison Wesley, 2003.

[GHM+03] Gudgin, M., Hadley, M., Mendelsohn, N. et al., *SOAP*, W3C, 1.2 edn., 2003, URL http://www.w3.org/TR/soap12.

[GK04] Graham Klyne, J. J. C., *Resource Description Framework (RDF): Concepts and Abstract Syntax*, W3C, 2004, URL http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

[HPSB+04] Harrocks, I., Patel-Schneider, P., Boley, H. et al., *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, DAML, 0.7 edn., 2004, URL http://www.daml.org/2004/11/fol/rules-all.

[HvHPS+01] Horrocks, I., van Harmelen, F., Patel-Schneider, P. et al., *DAML+OIL*, DAML, 2001, URL http://www.daml.org/2001/03/daml+oil-index.html.

[IBM02] IBM, BPWS4J, Online Resource http://www.aplhaworks.ib,.com/tech/bpws4j, 2002, URL http://www.aplhaworks.ib,.com/tech/bpws4j.

[JAKL+02] Jagadish, H., Al-Khalifa, S., Lakshmanan, L. et al., Timber: A native XML database, in *The VLDB Journal*, vol. 11, pp. 274–291, 2002, URL http://db.uwaterloo.ca/seminars/notes/timber.pdf.

[Jur04] Juric, M., *Business Process Execution Language for Web Services*, PACKT Publishing Ltd, 2004.

[KAP+02] Karvounarakis, G., Alexaki, S., Plexousakis, D. et al., RQL: a declarative query language for RDF., in *WWW*, pp. 592–603, 2002, URL www2002.org/presentations/christophides.pdf.

[Kay04] Kay, M., *XSL Transformations (XSLT) Version 2.0 - Working Draft*, W3C, 2004, URL http://www.w3.org/TR/2004/WD-xslt20-20041105/.

[KBS04] Krafzig, D., Banke, K. and Slama, D., *Enterprise SOA: Service Oriented Architectures Best Practices*, Prentice Hall, 2004.

[Kin04] King, N., Using XSL in XML View Management, *Technical Report ISG-04-04*, Dublin City University, Ireland, 2004, URL www.computing.dcu.ie/~isg/publcations/ISG-04-04.pdf.

[Kin05] King, N., The XLIM Architecture, *Technical Report ISG-05-01*, Dublin City University, Ireland, 2005, URL www.computing.dcu.ie/~isg/publcations/ISG-05-01.pdf.

[KKT+04] Kaykova, O., Kononenko, O., Terziyan, V. et al., Community Formation Scenarios in Peer-to-Peer Web Service Enviroments, in *IASTED International Conference on Databases and Applications (DBA 2004)*, pp. 62–67, ACTA Press, 2004, URL http://www.cs.jyu.fi/ai/papers/DBA-2004.doc.

[KMP+03] Karvounarakis, G., Magkanaraki, A., Plexousakis, D. et al., Querying the Semantic Web with RQL., in *Computer Networks*, vol. 42(5), pp. 617–640, 2003.

[MBD+03] Martin, D., Burstein, M., Denker, G. et al., *OWL-S 1.0 Release*, OWL-S Coalition., 2003, URL http://www.daml.org/services/owl-s/1.0/.

[MBH+04] Martin, D., Burstein, M., Hobbs, J. et al., *OWL-S: Semantic Markup for Web Services*, DAML, 2004, URL http://daml.org/services/owl-s/1.1/overview.

[Mei02] Meier, W., eXist: An Open Source Native XML Database., in *Web, Web-Services, and Database Systems*, pp. 169–183, 2002.

[MR04] Mark Roantree, Z. B., Noel King, Services for large P2P Networks, in *ERCIM Working Group for the Semantic Web Crete*, 2004, URL www.computing.dcu.ie/~nking.

[MSZ01] McIlraith, S., Son, T. C. and Zeng, H., Semantic Web Services, in *IEEE Intelligent Systems*, vol. 16(2), pp. 46–53, 2001, URL www.ksl.stanford.edu/people/sam/ieee01.pdf.

[NSS03] Nagappan, R., Skoczylas, R. and Sriganesh, R. P., *Developing Java Web Services*, Wiley Publishing Inc., 2003.

[OAS04] OASIS, UDDI Executive Overview: Enabling Service-Oriented Architecture, *Technical report*, OASIS, 2004, URL http://uddi.org/pubs/uddi-exec-wp.pdf.

[OBR05] O'Connor, M., Bellashène, Z. and Roantree, M., An Extended Preorder Index for Optimising XPath Expressions, in *Proceedings of the 3rd International XML Database Symposium (XSym)*, 2005, pending Publication.

[O'C04] O'Connor, M., Object-Relational Storage and Retrieval in Oracle XMLDB, *Technical Report ISG-04-02*, Dublin City University, Ireland, 2004, URL http://www.computing.dcu.ie/~isg/publcations/ISG-04-02.pdf.

[ORB05] O'Connor, M., Roantree, M. and Belleshene, Z., Level based Indexing for Optimising XPath Expressions, *Technical Report ISG-05-02*, Dublin City University, Ireland, 2005, URL www.computing.dcu.ie/~isg/publcations/ISG-05-02.pdf.

[PKY03] Papazoglou, M., Kramer, B. and Yang, J., Leveraging Web-Service and Peer-to-Peer Networks, in Elder, J. and Missikoff, M., eds., *Proceedings of Advanced Information Systems Engineering, 15th International Conference, CAiSE 2003*, pp. 485–501, 2003, URL infolab.uvt.nl/pub/papazogloump-2003-59.pdf.

[POS+04] Patil, A. A., Oundhakar, S. A., Sheth, A. P. et al., Meteor-s web service annotation framework, in *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pp. 553–562, ACM Press, New York, NY, USA, 2004, URL www2004.org/proceedings/docs/1p553.pdf.

[PSN+03] Paolicci, M., Sycara, K., Nishimura, T. et al., Using DAML-S for P2P Discovery, in *proceedings of the International Conference on Web Services, ICWS 2003*, 2003, URL http://www-2.cs.cmu.edu/softagents/papers/p2p_icws.pdf.

[PSS+04] Paolucci, M., Soudry, J., Srinivasan, N. et al., A Broker for OWL-S Web Services, in *AAAI Spring Symposium Series*, 2004, URL http://www.daml.ecs.soton.ac.uk/SSS-SWS04/40.pdf.

[RB04] Roantree, M. and Bellahsène, Z., Querying Distributed Data in a Super-peer based Architecture, in *15th International Conference on Database and Expert System Applications*, pp. 296–305, 2004.

[Sch01] Schöning, H., Tamino - A DBMS designed for XML., in *ICDE*, pp. 149–154, 2001, URL http://www.csd.uwo.ca/courses/CS853b/tamino.pdf.

[SCZ04] Solanki, M., Cau, A. and Zedan, H., Augmenting Semantic Web Service Description With Compositional Specification, in

World Wide Web Conference 2004, pp. 544–552, 2004, URL www.www2004.org/proceedings/docs/1p544.pdf.

[SF03] Schmauch, C. and Fellhauer, T., A Comparison of Database Approaches for Storing XML Documents, in *XML Data Management: Native XML and XML-Enabled Database Systems*, vol. 1, pp. 519–546, 2003.

[SPA+03] Sycara, K., Paolicci, M., Ankolekar, A. et al., Automated Discovery, Interaction and Composition of Semantic Web Services, in *Journal for Web Services*, vol. Vol. 1(1), pp. 27–46, 2003.

[SVS+03] Sivashanmugam, K., Verma, K., Sheth, A. et al., Adding Semantics to Web Services Standards, in *Proceedings of the 1st International Conference on Web Services (ICWS'03)*, pp. 395–401, 2003, URL lsdis.cs.uga.edu/lib/download/SVSM03-ICWS-final.pdf.

[SWM04] Smith, M., Welty, C. and McGuinness, D., *OWL Web Ontology Language Guide*, W3C, 2004, URL http://www.w3.org/TR/2004/REC-owl-guide-20040210/.

[TK03] Terziyan, V. and Kononehko, O., Semantic Web Enabled Web Services: State-of-Art and Industrial Challenges, in *International Conference on Web Services*, pp. 183–197, ICWS, 2003, URL old.netobjectdays.org/pdf/03/papers/icws/28530183.pdf.

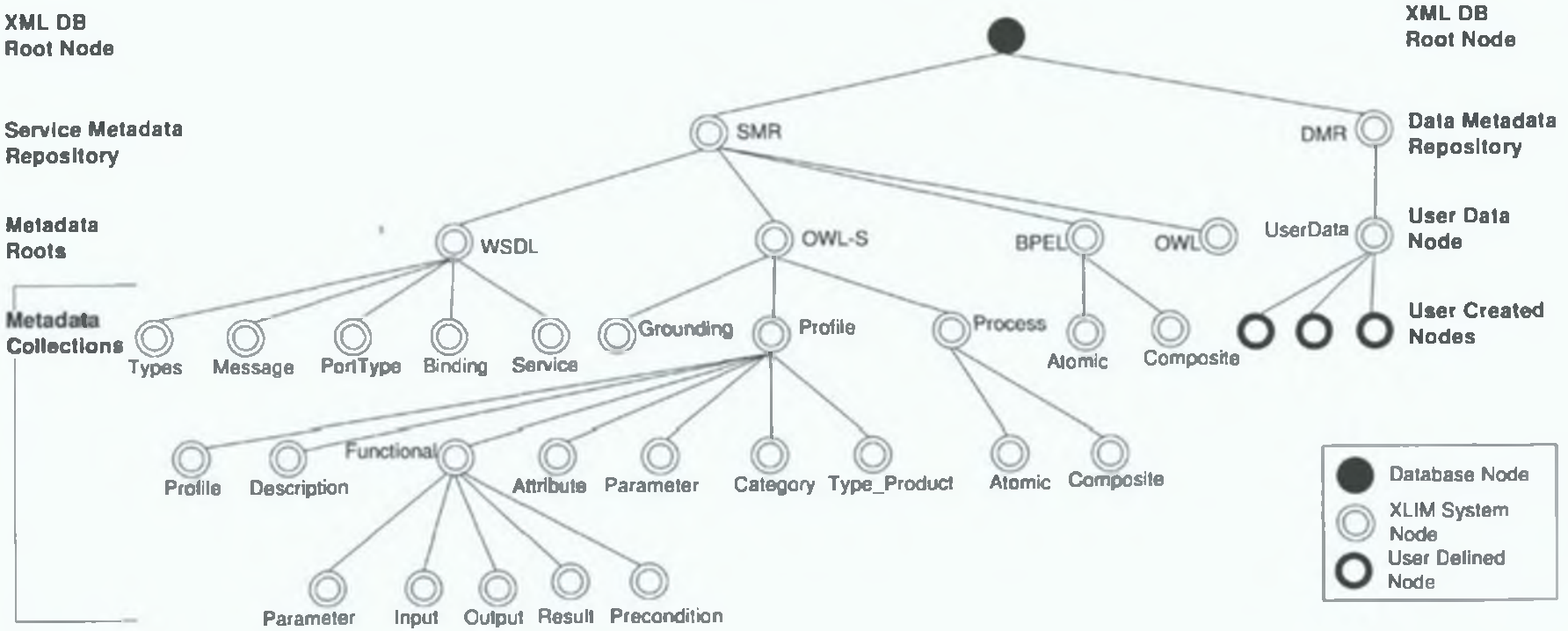
[TP04] Traverso, P. and Pistore, M., Automated Composition of Semantic Web Services into Executable Processes, in et al., S. M., ed., *International Semantic Web Conference*, pp. 380–394, 2004, URL http://sra.itc.it/people/traverso/papers/iswc04.pdf.

[VSS+04] Verma, K., Sivashanmugam, K., Sheth, A. et al., METEORS WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services, in *Journal of Information Technology and Management*, 2004.

[W3C04] W3C XML Schema Working Group, *XML Schema*, W3C, 2004, URL http://www.w3c.org/XML/Schema.

Appendix A

XLIM Storage Model for Services


XML DB
Root Node
Service Metadata
Repository
Metadata
Roots
Metadata
Collections
SMR
DMR
WSDL
OWL-S
BPEL
OWL
UserData
Types
Message
PortType
Binding
Service
Grounding
Profile
Process
Atomic
Composite
Profile
Description
Functional
Attribute
Parameter
Category
Type_Product
Atomic
Composite
Parameter
Input
Output
Result
Precondition
XML DB
Root Node
Data Metadata
Repository
User Data
Node
User Created
Nodes
Database Node
XLIM System
Node
User Delined
Node