CHAPTER 7
CONCLUSION AND FUTURE SCOPE

The Web Services Interoperability presents an effective means where by existing, perhaps loosely defined, system functionality can be adapted to operate in a web services paradigm. Through the use of Service Delegates, the details associated with directly interfacing with local system functionality are encapsulated and effectively isolated from reusable framework components. Semantic web presents design incorporates technologies such as inference engines, rule-based systems, web services and service-oriented architectures to provide the needed infrastructure to support meaningful interoperability among context-based systems.

In order to facilitate the interoperability, has developed a set of technologies, standards, and interface protocols, for interoperability of data, information, and systems over the web. The web service technology and standards are widely accepted and used by the for interoperability among grid systems.

The solution to interoperability presented in the research goes beyond traditional web services architectures by supporting the representational disparity typically exhibited by context-oriented systems. Rather than constraining interoperating systems to common representations, the interoperability bridge provides a mechanism for managing the potentially complex representational translation between interoperating systems.

The interoperability between the web and Grid technologies is promising in solving important challenges at different levels in the project architecture: data level, execution/processing level and portal level of web services.
The system explores an effectively integrated approach of grid technology and web services to implement the sharing and interoperability of multi-source spatial data, distributed on heterogeneous platforms.

This study suggests as a system bottom platform to organize the various services coming from heterogeneous environment. Resources management tools is used to support the deployment and management of web services; and Grid are used in distributed entire network. The architecture of heterogeneous systems based on the Web Services is put forward, through the module of virtual data warehouse for realizing data mapping and interoperability and the implement of architecture is limited to the product data of heterogeneous systems. study show this approach is a feasible approach to support a sharing and interoperability for multi-source data distributed in heterogeneous platforms.

Aggregation conflicts are caused by the concept Files of LMS1 designing the data concerning resource files, clustering them around each extension file. Similar conflicts occur in LMS2, these conflicts will be resolved by mapping rules that are shown in the table of rules. The study demonstrate and validate the semantic integration approach. The local ontology is developed using a well known tool, protege. Not only was the ontology construction implemented but also consistency checking of the integration process.

The main contribution of this work is the introduction and use of Web service to achieve interoperability in a heterogeneous medical environment. The implementation of it already been carried out. Web service acts as a distributed middleware to facilitate the interoperability of the whole system with the support of distributed technologies. An overall assessment was also executed to test how well to address the interoperability problem with the support of web service and distributed
technologies. This assessment consists of testing on applications developed in diverse programming languages, system platforms and database systems.

The healthcare information systems require fundamental restructuring to interoperate with new network centric environments. The main characteristic of these largely distributed systems is being *Network Centric* which refers to wide usage of network based applications and data communication using different web-based architectures. Such systems, usually use providers or technologies which restrict the wide usage of their internal services by their local community.

The new standards in healthcare domain provide advanced techniques that allow data and service identification and interoperability to be performed in a very systematic manner. This case study describes the process of designing and implementing an ontology related to data structure of CER, using as a standard reference the UMLS.

The use of UMLS Metathesarus was very useful to validate our ontology as well as the interviews with the medical specialists. This ontology build applications to acquire, store and analyze information that compose CER. The ontology can also be used to ensure semantic interoperability among systems in this domain.

The HIIS architecture provides a basis for the implementation of the experimental prototype for future research, and it could be implemented on a Health Grid application in the real environment to:

i. build up a HealthGrid-enabled application,

ii. establish the feasibility of semantic heterogeneity, and

iii. assess the HealthGrid implementation of the proposed architecture.

The HealthGrids are designed and used specifically for clinical use and medical studies, both representing areas where data integrity and platform compatibility are critical to the provision of consistent medical
information to the various stakeholders of healthcare. HIIS application architecture is designed to semantically integrate heterogeneous data resources.

7.1 Proposed Interoperability Architecture

The Interoperability View in order to capture specific reusability of communication protocol among systems, subsystems and/or components, avoiding the use of a Data transformation. The Interoperability takes advantage of well defined system use cases. The choice of activating services depends on the use case that the invoker’s event needs. The framework focuses on the logical and Interoperability Architectural views.

The proposed approach is built on the following insights

i. Monitoring the correct deployment of system requirements;
ii. Offering an effective interoperability capability
iii. easily applied to other components, subsystems or systems; and
iv. Providing self-adaptive services at run-time according to received external events.

The framework architecture is structured in three modular views. They represent the entire system based on different concern areas, which are:

i. Logical (functionalities),
ii. Interoperability (reusability), and
iii. Technical (infrastructure / feasibility).
The proposed architectural structure is

![System Architecture Diagram]

Figure 7.1: Architectural Structure
(Source compiled by Researcher)

The Logical View covers a business representation of the system, which is acquired through functional requirements specification. Framework is a “partially completed application” that customizes a specific application, and it is implemented by the Logical View.

i. Logical (functionalities),
ii. Interoperability (reusability), and
iii. Technical (infrastructure / feasibility).

The Technical View is concerned with non-functional requirements or Quality of Service (QoS), used to achieve as well as possible the functional aspects.
A Design Pattern is a generalized solution to a commonly problem occurring. The common pattern used for interoperability among components and subsystems during the design phase. The process of discovering a specific pattern in the Interoperability View is a design pattern that implements specific property that is not delivered by the analysis model. The design model differs from the analysis model in the way that it contains aspects that are not required but included to make the entire system to work better. The pattern is used to solve the interoperability issue among systems, subsystems and/or components.

7.2 The Framework

Frameworks allow code reuse and fast application developments and design patterns and frameworks are highly synergetic. This pattern can be used to describe a framework. The structure diagram is designed to plug-in CS services at run-time, based on external events.

![Framework Structure Diagram](Source:compiled by Researcher)

**Figure 7.2: Framework Structure**

The External and Internal Invokers Capsules have specifics state chart to implement their own behavior. The Framework structure has also a connection with interoperability design pattern. It is then applied on their
state chart by extending the component state which is responsible for the communication with the CS, during the design phase.

The researcher has suggested design patterns as a solution have been created to optimize the interoperability issue into a generalized solution. Therefore, the pattern is applied by inserting it in the specific state that exchange data with the CS. The figure shows a basic example of an Invoker Capsule and also shows the Interoperability Pattern applied to the state responsible to connect to the CS.

![State Diagram for service invoker](image)

Figure 7.3: State Diagram for service invoker
(Source:Compiled by researcher)

![Invoker Interoperability Pattern](image)

Figure 7.4: Invoker Interoperability Pattern
(Source compiled by Researcher)
The Interoperability Pattern applied to the Working state.

7.3 Web Service interoperability

Interoperability is the ability of communication, executing programs and exchanging data between functional cells in the case of that user know little or even nothing about characteristic of these functional cells. Two components: A and B are interoperable, if A can send requests P for services to B, based on a mutual understanding of P by A and B, and if B can similarly return mutually understandable responses Q to A.
Above interoperability works for the

i. Freely exchange all kinds of information about the objects; and

ii. Cooperatively, over networks, run software capable of manipulating such information.

7.4 Web Service Interoperability Model

Service interoperability is implemented by interaction. Service Interoperability Model includes

i. Service (service type),

ii. operation (operation type),

iii. interface (interface type),

iv. Message (message type), and

v. Binding.

Figure 7.7: Web Service interoperability
(Source: Compiled by researcher)
i. Service: Representation mode of integrity system, subsystem, component, module, processing stream and database etc.

ii. Action: (inside and outside the service) It implements information exchange between services. It is an action when an interface communicates with one or multiple services and its environment.

iii. Interface: Service Communication is implemented by interfaces. The interface connects service of system to provide capability of information exchanging.

iv. Binding: It is a mechanism that connect more than two interfaces to implement interoperability.

v. Operation: It is an abstract, nomenclative specification of a transformation or query that an object to execute.

vi. Message: It is an aggregation of character string that composed of text and numbers to record actions and events. It is an information format and content exchanged between services, and processing object of interaction.
7.5 Interoperable Mechanism for Web Services

Figure 7.8: Interoperable mechanism for web services
(Source: Compiled by researcher)
Interoperable mechanism consists with four parts:

i. Client based on browser,
ii. Web application server,
iii. application server and
iv. registry center.

i. **Client**:

   a. **Data Search**: client is used for sending request to background service or data service over Internet, and search or other data that needed to be processed or interoperated for processing of data on client, which implement generating of communication link.
   
   b. **SVG Display**: to display SVG data transferred from background service Map Display: to implement “thin client”, map displaying is to display image passed from background service (WMS operation) such as JPEG and so on.
   
   c. **Data Query**: to query features in image, find out further relative information such as area, length, name etc.
   
   d. **Data Edit**: to provide functions of deleting features, altering attributes and editing symbols etc. Data Extract: to Save data of query result, and generate corresponding file on client.
   
   Map Service Integration: to implement integration of WMS from different software.

ii. **Web Application Server**

   Web application Server a commercial application server, includes following parts as follows:

   a. **Request Classifier**: to classify HTTP requests of client, requests are classified to data request; basic view request; interoperability request.

   b. **SOAP Packer**: to encapsulate relative request according to request classifier by SOAP format. SOAP Analyzer: to parse response and
send result to client, according to SOAP response of background service.

iii. Application Server:

a. **SOAP Analyzer**: to parse request according to Web application server’s SOAP request, and send response result to corresponding components of application server to be processed.

b. **Data Search**: to response data request from client, and implement data service search on service registry center. Register on registry center is required.

c. **Aggregation of View Operation**: to implement basic view operation of, such as Zoom in, Zoom out, Cruise, Full Screen etc

d. **WMS Operation**: to provide base for view operation, each Zoom in and Zoom out operation need to submit request to WMS service.

e. **WFS Operation**: to support querying and processing feature on browsers as well as basic editing feature.

f. **Data Conversion**: to implement data conversion between different data sets and/or other data source. These data with different format shall be interoperated.

g. **Service Register**: to provide service description and registry mechanism to components that needed to register on registry center.

IV. Registry center:

Registry center is mainly to provide functions of implement service register to service providers that want to implement data and functional interoperability. It includes several parts as follows:

a. **Service Search**: to implement data and function service search, according to search request of GML application server.

b. **Service Register**: to provide WSDL description for spatial service and service register and storage mechanism.

c. **Cancel Service**: service provider may cancel registered service by cancel service function.

b. **Description Distribution**: response for exterior request for search.
V. **Exterior service**

It can implement integration and interoperability with exterior service, which process still depends on SOAP request to implement binding and activation of service.

### 7.6 FUTURE SCOPE

The Interoperability framework provides a high-level context and structure within which interoperability discussions can occur. It identifies key categories and issues that need to be addressed. It will not provide reference architecture and a high-level model that identifies the primary systems involved in the future “Smart Grid” along with key system boundaries, information exchanges and interactions. Such architecture and model is needed in the future.

The architecture would be based on the foundation that distributed generation, demand response and transmission/distribution technologies will become automated through the use of modern large-scale, distributed computing technology.

In order to fully evaluate the expected benefits of the proposed architecture, there is a need to implement this architecture in a real Grid environment, which then could be tested for evaluation of scalability and reliability. The architecture provides a basis for the implementation of the experimental prototype for future research, and it could be implemented on a Grid application in the real environment to develop a Grid-enabled application, demonstrate the feasibility of semantic heterogeneity, and evaluate the implementation of the proposed architecture.

There are many avenues that can be explored on the basis of this study. The architecture proposed has the potential to be widely applicable, but has yet to be proven for its functional feasibility when implemented in a real environment. Using a real environment for exchanging heterogeneous information, confirmed that the can act as a mapping tool to provide integration of databases. In distributed environment.
The further work is to find a way to make full use of Globus’s resource optimization ability to enhance the web service’s performance and efficiency. Metadata is increasingly used to define the ability to increase discovery reuse and to facilitate interoperability.

This analyses a number of possible approaches using Semantic Web standards and outlines how the chosen approach will extend my research in implementing an automated system for the integration of Web-based content.