ARC Linkage Project

Integration of Built and Natural Environmental Datasets within National SDI Initiatives

Final Year Student Project:
Geo Web Service Requirements for Spatial Data Interoperability

Jessica Deane
October 2006

Supervisors:
Dr. Abbas Rajabifard
Mr Hossein Mohammadi
Mr Andrew binns

Centre for SDIs and Land Administration
Department of Geomatics
The University of Melbourne
Geo Web Service Requirements for Spatial Data Interoperability

J.Deane
Department of Geomatics
The University of Melbourne
Victoria, AUSTRALIA 3010
j.deane2@ugrad.unimelb.edu.au

Abstract

Geo Web Services are set to play an increasingly important role in the technical and network developments of Spatial Data Infrastructures. Geo Web Services aim to provide a standard means of interoperation between different software applications running over the web. This paper outlines research conducted into the use of Geo Web Services software in promoting interoperability between distributed systems. This research documents the development of building a prototype in the chosen software. The subsequent results and discussion of how levels of interoperability are addressed in the context of Spatial Data Infrastructures is then given.
Table Of Contents

1.0 Introduction ........................................................................................................................................... 3
  1.1 Aim.......................................................................................................................................................... 4
  1.2 General Approach .................................................................................................................................. 4

2.0 Spatial Data Heterogeneity and Interoperation .............................................................................. 5
  2.1 Spatial Heterogeneity .......................................................................................................................... 5
  2.2 Metadata ............................................................................................................................................... 6

3.0 Open GIS and Distributed Spatial Resources ............................................................................. 6
  3.1 Open GIS ............................................................................................................................................. 6
  3.2 Distributed Technologies ................................................................................................................... 6
    3.2.1 XML ............................................................................................................................................... 6
    3.2.2 GML ............................................................................................................................................. 7
    3.2.3 Geo Spatial Web Services Interfaces .......................................................................................... 7
    3.3.4 Component Based Architectures ................................................................................................. 7

4.0 Developing a Prototype .................................................................................................................. 9
  4.1 Architecture ......................................................................................................................................... 9
  4.2 Software Evaluation ............................................................................................................................ 10
  4.3 Software Stack ................................................................................................................................... 10
    4.3.1 JTS Topology Suite ....................................................................................................................... 11
    4.3.2 GeoTools ...................................................................................................................................... 11
    4.3.3 GeoServer .................................................................................................................................... 11
    4.3.4 uDig ............................................................................................................................................. 11
    4.3.5 Eclipse ......................................................................................................................................... 12
  4.4 Setting up the Prototype ..................................................................................................................... 12
    4.4.1 Installing Application Environments .......................................................................................... 12
    4.4.2 Explore Capabilities of Software ................................................................................................. 12
    4.4.4 Design of the Prototype ............................................................................................................... 14
    4.4.5 Creating a Plug-In Project ........................................................................................................... 14
  4.5 Constructing a Prototype ................................................................................................................... 15

5.0 Results .................................................................................................................................................. 16
  5.1 Selecting Datasets ............................................................................................................................... 16
  5.2 Selecting Format .................................................................................................................................. 17
  5.3 Selecting Metadata .............................................................................................................................. 17
  5.4 Selecting Coordinate Reference System .......................................................................................... 17
  5.5 Report .................................................................................................................................................. 18
  5.6 Prototype Java Classes ....................................................................................................................... 19

6.0 Discussion ............................................................................................................................................ 20
  6.1 Technical Discussion ........................................................................................................................... 20
  6.2 General Discussion .............................................................................................................................. 21

7.0 Conclusion ........................................................................................................................................... 23
1.0 Introduction

In order for spatial information and services to be more useful and enhance decision making tasks, it is vital that such information and services be capable of interoperation in open and distributed environments. Spatial Data Infrastructures (SDI) have stemmed from the need to expand and distribute Geographic Information Systems (GIS) into cooperative environments (Nogueras-Iso et al 2005, Feeney and Williamson 2000). SDIs therefore ‘facilitate and coordinate the exchange and sharing of spatial data between stakeholders in the spatial data community’ (Rajabifard et al 2004). Such an infrastructure does not merely involve interacting technical and network components, but must also consider policies, standards, databases, metadata, resource providers, end users and institutional arrangements across different jurisdictional levels as depicted in Figure 1.

![Figure 1: A system view of the SDI components (Source: Nogueras-Iso et al 2005)](image)

The development of distributed or web based spatial services and open standards for data exchange, are driving forces for the technical and network components of SDIs (Nogeras-Iso et al 2005, Zaslavsky 2000). In particular, the success of the web as a global infrastructure is the momentum for delivering services over the web. The strengths of the web as an information distributor include that it is simple, ubiquitous and platform neutral. The web services movement therefore is about the fact that the advantages of the web as a platform cannot only apply to information but to services.

In order for systems to interact with web services, standards and interfaces have been developed to overcome the deficiency of the web’s HTML protocol, which is a human readable format only. Web services thus describe software systems that are designed to support machine-to-machine interaction over a network primarily by the use of XML technology, a machine readable format. The appeal of Web services architectures is that they provide a standard means of interoperation between different software applications, running on a variety of platforms.

Spatial information however presents additional challenges when employing web services for spatial resources. This is because spatial information has a fundamental visual component that must be considered when exchanging and using this information. Secondly the heterogeneous nature of spatial
data makes it complex because there is no universally accepted standard for its representation, and it has been designed by different groups of people for many different purposes (Fallman 2004).

1.1 Aim
Evolving standards for spatial data and services are the driving forces behind spatial data interoperability practices. There are three principle standardization bodies contributing to this field including The Open Geospatial Consortium (OGC), the International Standards Organization (ISO) and the World Wide Web Consortium (W3C).

The OGC is the most relevant to this research because it is facilitating the challenges of spatial data sharing and interoperability in open and distributed environments. In doing so, the OGC have developed a number of openly available spatial interface specifications and standards that promote data exchange and storage of spatial information. Among these standards, and of most importance to this research, include the Geography Markup Language (GML) and interfaces for Geo Web Services such as the Web Feature Server (WFS). These standards are set to play an increasingly crucial role in the future of distributed spatial services and development of SDIs (Chunithipaisan et al 2003, Nogueras-Iso et al 2005).

Increasingly these OGC specifications are being adopted and implemented by GIS vendors and users in order to provide solutions to the problems of web based spatial data dissemination (Chunithipaisan et al 2003). Among these vendors, many emerging Open Source GIS (OpenGIS) initiatives are creating products to fill every level of the OpenGIS SDI stack (Ramsey 2006). These open source projects are built on the premise that open source software allows for innovative and accelerated development among the GI (Geographic (spatial) Information) community, allowing for the true collaborative nature on which SDI initiatives are established (Holmes et al 2005).

It is therefore the aim of this project to look at the different levels in which interoperability need to be achieved for spatial data and resources in distributed web-based environments. A prototype will be developed in suitable OpenGIS software that implements OGC Geo Web Services specifications. This is in order to examine how some of the interoperability challenges are met by the use of Geo Web Services and their associated architectures in the context of SDI initiatives.

1.2 General Approach
In order to fulfill the above stated aim, the levels in which interoperability need to occur for spatial resources in distributed environments is examined. A simplified architectural model is created in order to understand the components and protocols involved in a Geo Web Services architecture. Software that implements the Geo Web Service standards and can act as a component in the architecture created was then chosen to develop a prototype application.
2.0 Spatial Data Heterogeneity and Interoperation

2.1 Spatial Heterogeneity

The definition of Interoperability offered by Bishr (1998) is ‘the ability of a system, or components of a system, to provide information portability and inter-application cooperative process control’. The main obstacle for the interoperation of systems is the different levels of heterogeneity in data and services managed by the systems (Nogueras-Iso et al 2005). Six levels of interoperability can exist, as depicted in Figure 2, between separate GISs in which a wide range of technology exists. These levels include the low level network protocols, hardware and operating systems, spatial data files, database systems, data models, and application semantics (Bishr 1998). The most challenging of these levels in terms of interoperability is the data model and application semantics levels. The goal of interoperability in SDIs is that these levels are made transparent to the end user in order to provide a single homogenous view of information from a variety of sources.

![Figure 2: Levels of Interoperability (Source: Bishr 1998)](image)

Due to the fact that GIS’s have been developed independently, several heterogeneity problems arise on the data model and application semantics levels where spatial data is represented. These heterogeneity problems can be classified into three types; syntactical, schematic, and semantic heterogeneity. Schematical heterogeneity refers to the schemata of different mental models, which occurs at the database design phase when classifying and assigning hierarchies to real world objects. Syntactical heterogeneity refers to the logical data model and its underlying DBMS, e.g., relational and object oriented, or the representation of spatial objects in the database (Bishr 1998). Semantic heterogeneity refers to the naming conflicts which occur in separate GI communities where the same names are given to different phenomena, or different names to the same phenomena.

The conceptual differences that result in semantic heterogeneity is one of the most serious limitations to interoperability and therefore has in recent times received the most attention. In order for the thematic attributes and geometric descriptions to be understood by outside users of a dataset, the metadata that describes these attributes and descriptions must be provided to the user. However this is not acceptable for achieving transparent interoperation when a number of datasets across a number of different databases and servers are used. However in order to achieve semantic interoperability from diverse
heterogeneous information sources, the meaning of the interchanged information has to be understood across the systems and therefore the metadata from each source must be used (Bishr 1998).

2.2 Metadata
In the development of SDI’s, it is essential to have metadata to appropriately document data and services. Metadata offers descriptions of content, quality, condition, authorship and any other characteristics of the resource. It constitutes the mechanism to characterize data and services in order to enable other users and applications to make use of such data and services. Catalog systems can be used to publish the metadata records of data or services. Thus, metadata and catalogs are the basic components that facilitate the accessibility and interoperability of the resources and services offered by a SDI (Nogueras-Iso et al 2005, Feeney and Williamson 2000).

3.0 Open GIS and Distributed Spatial Resources

3.1 Open GIS
“Open source” GIS or “OpenGIS” is GIS software in which the source code is available for modification and redistribution to the general public. OpenGIS encourages independent developers to make deeper contributions to projects, which in turn promotes growth due to a community of shared interest. This is argued to support the growth of SDIs in developing countries because it counteracts the negative balance of trade impact of a countries reliance on a few major software suppliers from wealthier countries (Holmes et al 2005). These countries therefore can use finances for SDIs for other crucial development activities involved such as capacity building and education. The diffusion of software tools to support SDIs are therefore only limited by the knowledge, learning and innovative energy of the users rather than the proprietary software rights or prices (Holmes et al 2005).

Currently there are two distinct developer groups of OpenGIS, those using a Java based development environment and those using a C based platform. However as standardisation practices have evolved and GIS has moved into the Web Services paradigm, tools and services are becoming available that are independent of either language (Ramsay 2006).

3.2 Distributed Technologies

3.2.1 XML
XML is the fundamental building block for distributed computing and web services. XML Web services are the platform allowing multiple disparate applications to interoperate regardless of the application environment or location. XML overcomes the limitations of HTML which is a markup language used to format human readable information presented to a web client after a request to a server. XML does not replace HTML, but works with HTML to facilitate the transfer, exchange and manipulation of data. XML is metalanguage allowing components on different computers to interact in a platform neutral and portable way. XML uses tags to characterize data rather than governing the display of data as in HTML (Hoffer et al 2005).

XML is the basis of more specialised languages used in Web services such as SOAP, WSDL and UDDI. Of most relevance to this research is SOAP (Simple Object Access Protocol) which allows web services
to exchange XML messages in a standard way over the HTTP protocol. XML is also the basis of the Geographic Markup Language (GML), a language designed specifically for spatial data. XML based standards such as these address the syntactical issues involved in integrating information from disparate sources (Thakkar et al 2001)

### 3.2.2 GML
GML has been developed by the OGC to foster data interoperability and exchange between different systems (Peng and Zhang 2004, Lake 2006). This is possible because GML uses XML to model, transport, and store spatial information in a standard way (OGC 2005a). GML is based on the OGC Abstract Specification that models the world in terms of features. This specification for modeling geography is a common one agreed to by the vast majority of all GIS vendors in the world (Lake 2006). GML schemata presents a standard way to describe features and their corresponding properties including coordinate reference systems, topology, time, units of measure etc (Peng and Zhang 2004).

### 3.2.3 Geo Spatial Web Services Interfaces
The development of services offered by SDIs has increasingly been guided by the web services movement (Nogueras-Iso et al 2005). This is because the standardised service interfaces such as the Web Feature Server (WFS) and Web Map Server (WMS) are breaking heterogeneity barriers between disparate spatial resources. This in turn makes combining or chaining sets of spatial resources located along distributed servers on a large scale possible (Morales 2006). A fundamental attribute of these services in terms of interoperability is that they are self describing. The descriptions that accompany these resources focus on exposing the behavior of the resource and its interaction mechanism or point of composition. These descriptions or metadata can then be made available through a catalog or registry with the use of standards such as UDDI.

This research will take a closer look at the OGC’s WFS Interface Specification that allows local or remote clients to retrieve geospatial data encoded in GML. The WFS allows client applications to access, query, create, update, and delete elements from the GML feature database server (Peng and Zhang 2004). The operations provided by the WFS include GetCapabilities, DescribeFeatureType, GetFeature, GetFeatureWithLock, GetGMLObject, LockFeature and Transaction.

### 3.3.4 Component Based Architectures
The OGC web services (OWS) Architecture is based on the Service-Oriented Architecture (SOA). SOAs are still developing and have been informally defined but for the purpose of this research they will be described as architectures in which reusable and functional components interact and provide services to each other that are independent of the application and the computing platforms on which they run. The advantage of this architecture is that it makes building and adjusting reusable component applications fast and easy.

The OWS architecture components provide services to clients, and/or other components. Multiple implementations of a component is called a service where as a single implementation is called a server (OGC 2005). Clients are software packages that provide access to the components or services to human users. Client software can either be thin, for example a web browser, a thick application, for example uDig, or chubby which is in between (OGC 2005). All components or services can be organised into four tiers as shown in Figure 3.
These tiers are categorized according to the role or service they provide in the architecture. The Information Management Services tier stores and provides access to data, the Processing Service tier provides services that can process data, and the Application Services tier contains services designed to support clients such as web browsers (OGC 2005). OGC standardised interfaces can exist on these tiers in order to promote interoperability between them.
4.0 Developing a Prototype

To develop a prototype based on service descriptions offered by the OGC, a simplified architectural model based on the OWS architecture previously shown was first constructed to prescribe the framework in which the prototype would operate. The analysis of suitable OpenGIS software as a component in this architecture could then take place. The model together with the software would then determine how a prototype could be created. The steps undergone in order to create a prototype are shown in the following diagram.

4.1 Architecture

A simplified architectural model was thus conceptualized so as to prescribe the components and protocols involved in a OWS architecture. Since many frameworks for implementing the OWS architecture exist, this was necessary in order to narrow the focus to spatial data at the feature level. The WFS is the specification for serving feature data encoded in GML as mentioned previously. The following model was thus created based on the OWS architecture and current interoperable standards:
Figure 5: Architectural Model prescribing necessary components for research

4.2 Software Evaluation
Due to a number of OpenGIS software currently being available, the most suitable for developing a prototype had to be considered. As the requirements for distributed spatial resources in an OWS architecture were discovered from creating a model in the previous step, a number of ideal software characteristics could be devised and compared against available application software.

The primary considerations for the software were the development language and software functionality. Due to having no experience in C programming it was necessary to find software written for Java or a web based language. It was also necessary that the software support current spatial interoperability standards covered in the model previously created. Therefore the software had to be capable of interacting with WFS and supporting GML. A number of application software suited these characteristics such as Deegree, JUMP, OpenMap, MapBuilder, GeoServer and uDig.

The next important factor was the ability of the software to be easily extended and developed. This meant that the software had to be comprised of a component based or modular framework, where components could be easily reused. Additionally it also meant that the software be well documented so that a third party developer could easily install and understand the software from a development perspective. The outstanding choice here was that of uDig and GeoServer, that are built from the GeoTools library using interacting plug-in components. This software also met the well documented requirement, as sufficient online materials were available for installing and developing this software.

4.3 Software Stack
The following section describes the software, the supporting libraries and the software dependencies in a software stack. The diagram below depicts these dependencies:
4.3.1 JTS Topology Suite
The JTS Topology Suite (JTS) is a Java Application Programming Interface (API) that follows the OpenGIS Simple Features Specification for SQL. This specification defines a standard SQL schema that supports storage, retrieval, query and update of simple spatial feature collections. JTS provides methods supporting spatial analysis on features restricted to 2D geometry, such as lines and polygons.

4.3.2 GeoTools
GeoTools is an open source, Java GIS toolkit for developing OpenGIS compliant solutions. The GeoTool’s feature model is provided by the JTS Topology Suite. GeoTools implements OGC specifications and other relevant standards as they are developed. Therefore GeoTools focuses on being a Java Code Library which can be used by servers such as GeoServer or desktop applications such as uDig.

4.3.3 GeoServer
GeoServer is a Java implementation of the OGC’s WFS specification. GeoServer is built on top of the GeoTools library. The GeoServer framework is a fully transactional WFS that supports a number of formats such as shapefiles and oracle but in particular PostGIS. GeoServer is able to be extended through the Eclipse platform.

4.3.4 uDig
uDig (User-friendly Desktop Internet GIS) is a spatial data viewer and editor with emphasis on the OpenGIS standards for Internet GIS, the WMS and WFS standards. uDig builds on the design, data structures and standards of GeoTools with the Rich Client Platform (RCP) of Eclipse to provide a common platform for building spatial application with open source components. In this way uDig extends the Eclipse framework in a standardised manner by way of adding functionality though creating new plug-ins.
4.3.5 Eclipse

Eclipse is an extensible, open source Integrated Development Environment (IDE) which consists of a collection of interacting components called plug-ins. Eclipse provides the extensible framework, tools and runtimes for developing and extending software applications such as uDig.

4.4 Setting up the Prototype

4.4.1 Installing Application Environments

In addition to the GeoServer, uDig and Eclipse software, the environment for these applications had to be installed to allow these applications to be configured and developed. This meant installing the relevant Java Software Development Kit (SDK) and Java Runtime Environment (JRE). As GeoServer is run through Internet Explorer using the local host ‘http://localhost:8080/geoserver’ the Microsoft Internet and Information Services (IIS) also needed to be installed for GeoServer.

4.4.2 Explore Capabilities of Software

With the architectural model created in section 4.1 in mind, the capability of the software in acting as a component in this architecture was explored. Being a fully transactional WFS, GeoServer was tested to see whether it could expose its information to a client and in turn accept WFS feature data from a remote server. A shapefile of Australian point data, called ‘atsic_regions_2001_point’, was used to create a feature type able to be served by the GeoServer WFS. The ‘FeatureType Editor’ in GeoServer automatically collected feature information for this data such as spatial reference system, geometry type and bounding box as can be seen in Figure 7.

![Figure 7: GeoServer Feature Type Editor for adding a shapefile feature type](image)

The necessity of clients capable of retrieving, viewing, and manipulating GML data served by the WFS then presented itself. Using Internet Explorer to act as a client, a getFeature request was made to the WFS to retrieve GML data as shown in Figure 8. A chubby client was then used in order to visualize the data using the OGC compliant MapBuilder that also runs through the web browser. A thick client, uDig, was then used to manipulate the data from the WFS. Geometry editing tools were used to remove points...
within the Australian border in the ‘atsic_regions_2001_point’ dataset, while some were added outside. These changes were then saved and run through the WFS again to see if changes to the WFS dataset were possible. This can be seen in Figure 9a and 9b.

As a thick client in the architectural model uDig was tested to see if it could access data from a remote WFS. The URL for a WFS getFeature request was therefore entered for a Canadian WFS server and added with local shapefile data as shown below. The feature enclosed by the red circle shows the external Canadian data.
4.4.4 Design of the Prototype

Since the focus of this research was investigating interoperability and integration capabilities offered by OpenGIS software when datasets are combined, it was decided that the prototype application would take on the form of an Integration Tool. This Integration Tool would collect different characteristics of spatial datasets and in particular the metadata.

Using Eclipse to develop uDig it was now suitable to design a Graphical User Interface (GUI) to collect datasets and their properties in order to process their ability for interoperation. It was an early design choice that the characteristics of the data needed were format, projection, coordinate reference system (CRS), metadata standards and the metadata file associated with datasets. In this way preprocessing could occur before datasets were combined.

4.4.5 Creating a Plug-In Project

Plug-Ins are structured bundles of code and/or data that contribute to a system. The Eclipse IDE consists of a collection of interacting plug-ins. These plug-ins are used to create uDig and other desktop applications. The function that plug-ins contribute to uDig can take the form of code libraries (Java classes with public API’s), platform extensions, or even documentation.

To add extra functionality to uDig for developing a prototype a Plug-in project had to be created using Eclipse. This first involved setting up and creating a feature editor plug-in whereby relevant folders were created to store source code, generated schema documentation, java/class files etc which was called ‘net.refractions.tutorials.integration’.

The next steps involved configuring the newly created plug-in. Two dependencies this new plug-in had on other plug-ins was then specified. The ‘net.refractions.udig.libs’ was the first dependency added which is basically the ‘glue between uDig and the rest of the open source world’. This dependency allows uDig to access and use the GeoTools code. The second dependency called ‘net.refractions.udig.project.ui’ is used for developing on the project view and user interface in uDig.

The last step in configuring the plug-in was to define an extension. The extension used was ‘net.refractions.udig.project.ui.tool’, which allows developers to make new tools for uDig. More
specifically the ActionTool of this extension was used. ActionTools implement the ActionTool interface, in order to create fire tools which perform a single action when selected in uDig.

4.5 Constructing a Prototype
The extended ActionTool class created in the last section was now used to begin to develop the Integration Tool prototype. Due to the need for this tool to accept input and datasets from a user, a small GUI was required. To do this, the Standard Widget Toolkit (SWT) from the Eclipse API platform had to be used in order to populate the GUI with widgets or graphical interface elements. A set of public available APIs known as javadocs, were accessed in order to find appropriate classes and methods to import and extend from here onwards. These API specifications exist for the Eclipse platform, and for the uDig and GeoTools frameworks.
5.0 Results

The integration tool developed in this project is useful as a sample prototype for comparing characteristics of data and their capability of integration. This tool allows entry of two datasets and some corresponding characteristics, namely format, CRS and metadata. This data is set to a 2D array, or table, and then compared. This flow of events is depicted in the following diagram:

![Integration Tool Process Diagram]

Figure 11: Integration Tool Process

5.1 Selecting Datasets

The first user input that the integration tool accepts is two datasets from the file system. As a number of limited formats are supported by uDig, an error routine is implemented only allowing valid data formats to be entered. If the format and number of datasets is correct then the user can continue to the next step by selecting ‘Next..’. This step is shown in the figure 12.
5.2 Selecting Format

The next input the user sees is the format information. This information is automatically retrieved from the datasets and set to the corresponding fields. The user is also able to manually select the format from a list. Formats for both datasets must be entered before a user can proceed to the next step. This can be seen in Figure 13.

5.3 Selecting Metadata

The next information to be gathered is the metadata information of both the datasets. This includes allowing the user to upload a file of the metadata that accompanies the dataset as well as the metadata standard used. This can also be seen in Figure 13.

5.4 Selecting Coordinate Reference System

The coordinate reference system was the next information to be input from the user. A list of predefined CRSs were listed for the user to select from.
5.5 Report
Once the above steps have been completed, the input data is sent to a service class. This class accepts the parameters passed to it then sets them into a 2D array as depicted in Figure 11. A series of statements are then executed comparing the characteristics of the two datasets. A report is then displayed to the user comparing the characteristics of the data and the capability of interoperability as shown below.

Figure 13: Final step in Integration Tool as shown to the user
5.6 Prototype Java Classes

Two Java classes were built in order to retrieve and correlate dataset information for two spatial datasets. The client class was the GUI class that implemented the ActionTool interface in order to appear under the ‘Tool Menu’ in uDig. The service class was the class responsible for setting data to the 2D array and performing the processing. The following Figure is a simplified representation of these two classes in a UML Diagram.

Figure 14: UML model of simplified prototype classes
6.0 Discussion

The following discussion will be broken into two parts. The first section will discuss some of the finer technical details of the research and future suggestions for implementing the prototype application in uDig or like application. The second section will take a broader perspective on the research conducted and will discuss interoperability capabilities offered by Geo Web Services in the context of SDI initiatives.

6.1 Technical Discussion

GeoServer was able to fulfill its role acting as a WFS in the architectural model by retrieving feature data and responding to a clients request for features encoded in GML. Because it was not an existing option in GeoServer to interact with external WFSs, development of this functionality would have been necessary in order to use GeoServer to develop a prototype. However due to GeoServer’s architecture being more of an application than a modularized framework, adding functionality would have been difficult and thus avoided. uDig on the other hand is comprised of an extensible framework consisting of reusable plug-in components with the ability to interact with external servers therefore making it the choice for developing a prototype.

uDig provides the ability to extend the Eclipse framework and customize its GIS capabilities using the modular java classes and GeoTools library that the application comes bundled with. The supported libraries coupled with the ability to access and manipulate WFS and WMS data, makes uDig a powerful tool when acting as a client in a Geo Web Service architecture. The ability to customise this application and add spatial or GIS related functionality however requires a thorough knowledge in Java feature implementations and in the GeoTools code library that implements OGC standards specifications. Therefore as a sample prototype application, the Integration Tool was able to manually retrieve dataset characteristics and correlate results. Further functionality by the use of known java classes can however be suggested.

The sample prototype application created in uDig if implemented in an OpenGIS client program such as uDig would need to be capable of more automatic dataset retrieval and transformation capability. uDig Java classes that act as a handle on local and external spatial resources need to be used for this functionality. The Catalog Plug-in for uDig is made up of the classes that handle management of spatial information. The IGeoResource interface in this Catalog Plug-In represents a handle on spatial information such as a shapefile, a feature made available through WFS, or a table in a spatial database. Feature classes from GeoTools used in uDig are used to represent these features and able to extract feature information. For example the FeatureSource interface used to represent external features can retrieve the features, bounding box and schema of the spatial data source. Before adding any spatial data as a layer to a map they need to be represented by such an interface and converted into a list of type IGeoResource. Java methods in these classes and interfaces would then be able to grab some of the characteristics of the datasets in order to produce the report to the user.

Although a predefined list of CRSs were used in the prototype, uDig and GeoTools use the EPSG Geodetic Parameter Dataset. The EPSG dataset provides CRS definitions, transformation and conversion parameters known as coordinate operations that can be downloaded from the EPSG website. GeoTools’ classes have been created to access this dataset and retrieve EPSG CRSs and reference them via a set of
unique codes assigned to each CRS. Additionally transformation parameters are used for coordinate transformations. A useful extension of the prototype would therefore be using these EPSG CRSs for the user to select from, or automatically retrieved from the input dataset. Transformation can then occur where possible using java classes that implement the EPSG transformation parameters.

Additionally for more meaningful information to be provided to the user the prototype could be extended to retrieve scale and bounding box information. This information could be used to inform the user when spatial datasets have such diverse areas and scale that combination would be meaningless. Currently uDig allows layers to be added to a map from diverse parts of the world, for instance a Canadian bird map and an Australian point data collection was added and could be barely seen as the map required extensive zooming out. Bounding box information can be retrieved in uDig using the IGeoResource interface.

The IGeoResource interface is able to act as a metadata accessor for spatial data resources. Therefore once retrieving metadata information through the Integration Tool this class would be able to retrieve information about resources such as the resource description, the bounding box and CRS if they exist. This solution would be limited however because uDig only provide support for the Dublin Core metadata standard.

While developing the code using the ActionTool interface some restrictions were encountered due to its underlying functionality. It was ideal that the prototype take on the form of a wizard like other functionality in uDig that accepts data from a user and/or adds layers to a map. However some limitations were encountered when trying to implement a wizard class. This occurred due to conflicting ‘run’ methods required by both the Wizard and ActionTool classes. The ‘DataWizard’ extension rather than the ‘ActionTool’ would have been more preferable for developing this prototype however developing from this extension was not possible.

6.2 General Discussion
Introduced by Rajabifard et al (2000) is the model of an SDI hierarchy in which political/administrative entities exist, formed by the integration of spatial datasets from the local level upwards by the integration of spatial datasets originally developed for the use in corporations operating at that level or below. SDIs therefore require that data is reusable and exchanged between components in a simple, cost effect and consistent way. This hierarchy is shown in figure below.

![SDI Hierarchy Diagram](source: Rajabifard et al 2000)
Component based Geo Web Service architectures or SOAs facilitate this model on a technical and network level so well because they allow computer systems operating across and in between the different levels to expose their services and resources for interoperation regardless of the computing environment which exists at each location. This component based approach breaks the dependency cycle which exists in computer systems whereby on top of the hardware at the lowest level sits the Operating system and APIs, on top of this level sits applications, next the development environment, and then on top the enterprise level applications. These levels in a computer system all strictly depend on the services provided to them from the levels below. Web service component based approaches therefore move away from these restrictions by allowing applications to interoperate regardless of their underlying platforms by using open web based standards. Moreover, component based Web service architectures take the advantages of the simplicity, ubiquity and availability of the web as a platform and apply it to services and resources that exist across the web.

In section two it was shown that interoperability needs to be achieved on a number of levels in order for SDIs to provide a single homogenous view of information from a variety of sources. By using uDig and GeoServer as a means of implementing current Geo Web Service and open standards, it was seen that interoperation on a number of these levels occurred. GeoServer was able to transform shapefile data into a feature collection encoded in GML, and made accessible to client applications through the WFS interface. uDig was able to access and manipulate data from a remote source and then also able to expose this information again through the local GeoServer WFS. Therefore interoperation occurred here across the network protocols, the DBMS, the hardware, the OS and to an extent the spatial data files.

SDIs frequently require that resources downloaded from the infrastructure are incorporated in a GIS and must comply with the same constraints as other layers wished to be added to a map (Nogueras-Iso 2005). Therefore spatial resources often require some preprocessing before being delivered to the user in order for interoperability to occur on the level of the spatial data files. The prototype was thus then created as an Integration Tool to address this level of interoperability by accepting some characteristics of data for preprocessing. This tool then was able to correlate information and present results to the user. Once information is retrieved, SDIs should also allow automatic transformations such as CRS transformation from data retrieved from the spatial data file, and therefore automatically update metadata information (Nogueras-Iso 2005). As discussed in the last section, the sample application prototype can be extended to add transformation functionality through the use of publicly available java API classes for uDig and GeoTools. Additionally uDig is able to automatically update manipulated data as was shown in Figures 9a and 9b.

The next levels of interoperability to be achieved in order for a transparent view to be provided by the SDI are the data model and the application semantics levels. The intention of achieving semantic interoperability from these levels is to provide seamless communication between remote systems without having prior knowledge of their underlying semantics (Bishr 1998). These levels as discussed in section two are the most challenging in which to achieve interoperability. The prototype touched on the issue of semantic interoperability by providing a means to collect metadata information. This is because the use of metadata describing data facilitates semantics interoperability because it promotes a commonly understood set of descriptors across systems and datasets. Nevertheless, one may also find heterogeneity in the schemas used for differing metadata standards (Nogueras-Iso 2005). uDig as discussed only provides support for the Dublin Core metadata standard and therefore is only a niche
solution to the problems of metadata heterogeneity. As Geo Web Services operate over the web and therefore on a global scale the possibilities for diverse metadata descriptions are exhaustive.

Machine understanding of metadata descriptions which conform to schemas from different domains is therefore a fundamental requirement for access to information within SDIs (Nogueras-Iso et al 2005). Although the XML standard for exchange of metadata has been established, the crosswalks, or mappings between related standards are still evolving. The current solutions to handle the problem of metadata interoperability may be classified into two main approaches: solutions that are based on the use of ontologies, and the creation of specific crossroads for one-to-one mapping (Nogueras-Iso et al 2005). Further discussion into these approaches are beyond the scope of this research, but are however critical for interoperability on the application semantics and data model level to be achieved.

7.0 Conclusion

OpenGIS software provide the means by which OGC Geo Web Service specifications can be implemented to provide solutions to spatial data dissemination in distributed systems. By selecting suitable OpenGIS software to develop and evaluate, an investigation was able to occur into how such specifications address the challenges of interoperability in open systems. The usefulness and applicability of Geo Web Services in meeting the challenges of interoperability could then be seen by placing them in the context of SDI initiatives.

A more thorough investigation was achieved by looking at the different levels of interoperability that exist between systems as proposed by Bishr (1998). It could then be seen that Geo Web Services are able to address interoperability on most of the levels in particular the network protocol level, the hardware & OS level and the DBMS level. The prototype application was developed in order to propose a solution to better address interoperability challenges on the level of the spatial data file. Further implementation of this prototype was then discussed in order to be able to better deal with interoperability on this level. The prototype also touched on interoperability on the level of application semantics by providing a means to capture metadata. The issues of interoperability on the data model and application semantics levels however still need to be addressed.
References


Peng, Z., and Zhang, C. (2004) ‘The roles of geography markup language (GML), scalable vector graphics (SVG), and Web feature service(WFS) specifications in the development of Internet geographic


