Master Thesis

submitted within the UNIGIS MSc programme
at the Centre for GeoInformatics (Z_GIS)
Salzburg University

The Feasibility of Building a Low Cost Client/Server Geographic Information System for Use in the Developing World

A Case Study in Kyrgyzstan

by

Andrew Smith

UP40102

A thesis submitted in partial fulfilment of the requirements of
the degree of
Master of Science (Geographical Information Science & Systems) – MSc
(GISc)

Advisor: Adrijana Carr

Bishkek, Kyrgyzstan, September 2008

Science Pledge

By my signature below, I certify that my thesis is entirely the result of my own work. I have cited all sources I have used in the thesis and have always indicated their origin.

(Bishkek – Kyrgyzstan, 30 September 2008)

Abstract

Fuelled by a growing awareness of Geographic Information Science among scientific and education professionals in Kyrgyzstan, is a desire to use the full potential of Geographic Information Systems (GIS) to carry out their work more effectively. Unfortunately, this desire has often manifested itself in the ad-hoc gathering of data sets which are kept inaccessible to the wider community, for both technical and political reasons. Ideally these data, along with a base of quality topographic and cadastral data, should be available via the internet for the benefit of all Kyrgyzstan. It is desirable that the cost of making these data available not be prohibitively high, recognising that Kyrgyzstan has an emerging economy and is limited in what it can pay.

GIS is a *system* - involving hardware, software, data, people and methodologies all working together to achieve a beneficial result to an organisation, community or country. Such a system works best when quality data is readily available to all stakeholders, but to make these data available requires the careful implementation of standards so that all involved know what to expect when accessing the data. Also it needs to be guaranteed that the accessing of these data is managed in such a way that the data's integrity is not compromised. There are numerous platforms available on which to build a GIS, many of which involve commercial software with high price tags and some tend to lock users into specific data formats and/or specific methods, which are not always transferable to other situations, systems or platforms. Because of the threefold need: for standards, for open access and for low cost, two terms frequently reoccur in this thesis – they are *Open Source* and *Open GIS*. The first, Open Source, refers to a way of developing software which has reached such a level of maturity that it challenges to the very core the commercial market driven model. The second term, Open GIS, refers to the work of the Open Geospatial Consortium (OGC) in developing standards for the exchange of geographic data over the internet.

This thesis explores the feasibility of building a *low cost* Client/Server GIS at the Kyrgyz State University of Construction, Transport and Architecture (KSUCTA) to serve geographic data to both an internal network of users via a local area network, and to a wider community over the internet. It is, essentially, a technical study, dealing with technical questions of how to implement such a system, as opposed to a political one, which would deal with questions of what data, if any, should be shared on the internet.

Acknowledgements

Firstly, to Professor Josef Strobl and the staff at UNIGIS Salzburg who do such an amazing job in facilitating the distance learning program for the MSc (GISc). Their level of commitment and academic professionalism will remain an inspiration to me.

Secondly, I would like to acknowledge my colleagues at the department of Geodesy and Geoinformatics at KSUCTA, especially Dr Ainura Nazarkulova and Mr Akjol Djenaliev who, under the leadership of Dr Akylbek Chymyrov, have supported me throughout the duration of this study by lowering my own teaching load and providing a continuous stream of professional feedback.

For checking and proof reading, I am very grateful to Dr J Keir Howard for his meticulous attention to the professional writing norms, and the many idiosyncrasies of the English language. Also, a huge thank you goes to Dr Adrijana Carr for her tremendous patience, support, feedback and advice throughout the duration of study.

My biggest thanks goes to my family, my wife Jackie and daughters Ayesha and Naomi, for standing beside me through the long, and often grumpy hours that went into producing this thesis – I hope now I can repay some of this time.

Table of Contents

List of Figures

List of Tables

Listing of File and Code Segments

List of Terms and Abbreviations

ASCII	–	American Standard Code for Information Interchange
CSW	–	Catalogue Service for Web [OGC]
DBMS	–	Database Management System
DCW	–	Digital Chart of the World
DTM	–	Digital Terrain Model
ESRI	–	Environmental Systems Research Institute (the creators of the ArcGIS suit of software)
GI	–	Geographic Information
GIS	–	Geographic Information Systems
GK	–	Gauss Kruger
GML	–	Geography Mark-Up Language
GUI	–	Graphical User Interface
HTML	–	Hypertext Mark-up Language
IC ISM	–	Intelligence Community Intelligence Security Marking [OGC]
INSPIRE	–	Infrastructure for Spatial Information in Europe
KSUCTA	–	Kyrgyz State University of Construction, Transport and Architecture
map server	–	is the generic name given to an application that takes GI data from a database or data store and serves to a client application. This should not be confused with MapServer which is a specific map or web serving application developed for the University of Minnesota
MNDNR	–	Minnesota Department of Natural Resources
MS4W	–	refers to the "MapServer for Windows" build of MapServer
NASA	–	National Aeronautics and Space Administration
NSDI	–	National Spatial Data Infrastructure
ODBC	–	Open Database Connectivity
OGC	–	Open Geospatial Consortium – also see [OGC]
OLS	–	Open Location Service (Geo–coding) [OGC]
OM	–	Observations and Measurements [OGC]
OWS	–	OpenGIS Web Services [OGC]
SDBMS	–	Spatial Database Management System
SDI	–	Spatial Data Infrastructure
SE	–	Symbology Encoding [OGC]

SensorML	–	Sensor Model Language [OGC]
SLD	–	Styled Layer Descriptor [OGC]
SOS	–	Sensor Observation Service [OGC]
SPS	–	Sensor Planning Service [OGC]
SQL	–	Structured Query Language
SRTM	–	Shuttle Radar Topographic Mission
SWE	–	Sensor Web Enablement [OGC]
TM	–	Thematic Mapper
TML	–	Transducer Mark–up Language (for transporting sensor data) [OGC]
UMN	–	University of Minnesota
USGS	–	United States Geological Survey
UTM	–	Universal Transverse Mercator
WCS	–	Web Coverage Service [OGC]
WFS	–	Web Feature Service [OGC]
WMS	–	Web Mapping Service [OGC]
XML	–	Extendable Mark-up Language

NB [OGC] means that the abbreviation is for an OGC specification

CHAPTER 1 INTRODUCTION

1.1 Motivation

KSUCTA has only in the last two to three years begun to offer courses in Geographic Information Systems (GIS), as part of its newly introduced programme in Geodesy and Geo-informatics. A result of the introduction of GIS is the creation and accumulation of geographic data. These data need to be stored on a centralised data server, i.e. a secure database that should be centrally maintained and accessed remotely.

The concept of remote access to GIS data is particularly relevant to the situation at KSUCTA, because it has many different sites around the city of Bishkek and the whole of Kyrgyzstan. Many people working at these sites are beginning to see the potential of GIS and some are actively working at developing databases. As there is no wide area network linking these sites, the internet offers an ideal way of accessing these data remotely. Additionally, as the data may well be made available to the general public over the internet, it is desirable that OGC standards be used when interfacing to the data.[1] Of particular importance in this is inclusion of metadata.

The beneficiaries of a centralised GIS data server would, initially, be those working with GIS at KSUCTA. This includes not only those specifically studying GIS, but also those people working in the many disciplines that use it, e.g. environmental engineers, land valuation assessors, water engineers etc. Whilst these people would all work with their own data, they would also be able to access other data, as well as being able to share their data without needing to copy them between the sites. For example, a department that deals with the construction of tunnels would be able to store engineering data related to specific a geological survey of a tunnel while overlaying roads and general geological data from the central database, as well as allowing their

1 All of KSUCTA's GIS data is currently stored in either ESRI shapefile or personal Geodatabase format, as most of the work done has been in ArcGIS at the university's GIS lab. However, future projects will require data to be accessible to other applications and, therefore, it is desirable to move away from a single vendor dependant format.

geological survey to be viewed by other users. In addition, the wider public would also benefit from the system via a pre-designed webpage interface.

1.2 Task description

Although it is recognised that it is highly desirable for KSUCTA to have a centralised client/server GIS, the task of this study is to *investigate the feasibility* of such a system rather than to build it. While a working GIS with a centralised data server (sometimes referred to as a map server) may well result from this study, of more importance is the process of assessing needs and investigating what is actually possible.

Therefore, the major question that this thesis will seek to answer is "Is it feasible to build a low cost client/server GIS that will fulfil the needs of KSUCTA? And if so, what is the underlying architecture for such a system?"

1.3 Approach

As KSUCTA has no existing spatial data infrastructure it is possible to use a purist "top down" approach when looking at this task. In other words, because there are no existing spatial databases, very little GIS software usage, and no GI systems to speak of, it is possible start with the big picture, knowing that there are very few internal constraints that need to be adhered to. This means not needing to build a map server that is only able to read an ArcInfo Librarian® database, for example, and then is only able to output to a specific type of client. But rather designing a system that is flexible and open to current and foreseen data transfer protocols and standards.

1.3.1 Theory

Visually the theory of a client/server GIS can be summed up in the diagram in Figure 1 below. Here it can be seen that the data server is *the* link between the database and *all* who want to use it. Furthermore, two important points can be seen in this diagram:

- Firstly – the data server is the *only link* between the database and the users, i.e. there is no back door for hackers and pseudo system-superusers to access the database, and
- Secondly – the flow of data is bi-directional, i.e. it is not just a viewing or data publishing interface, but should facilitate both the querying and updating of existing data, and the creation of new data.

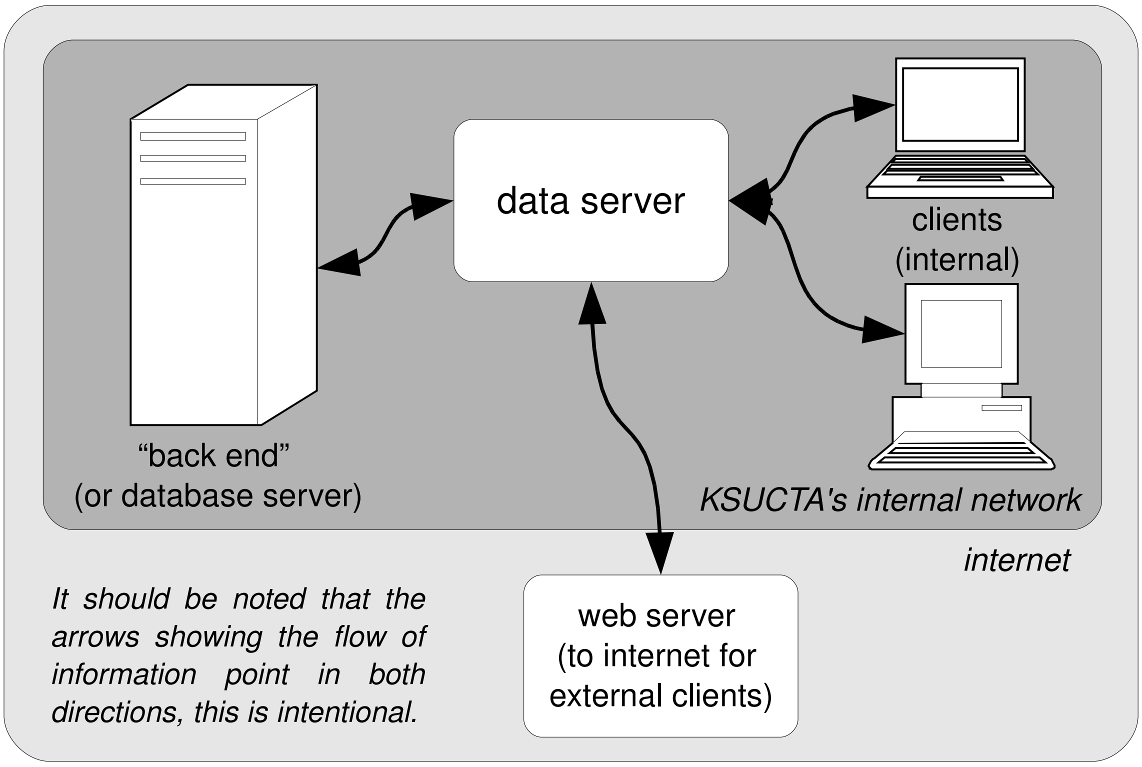

data server
clients
(internal)
"back end"
(or database server)
KSUCTA's internal network
internet
It should be noted that the arrows showing the flow of information point in both directions, this is intentional.
web server
(to internet for
external clients)

Figure 1: A conceptual diagram showing data flows through the data server.

Traditionally, geographic information systems that use a client/server architecture place the data server in the centre between the *back end* database and the clients. Thus all access to the database goes through the data server. In all probability this tradition will be followed. However, what will be unique in this study will be KSUCTA's specific data, the list of potential clients, and the use of low cost (i.e. open source) technology.

1.3.2 Methodology

It is actually to KSUCTA's advantage that it is only now starting out in GIS, as there has been so much development already completed that can be benefited from without needing to “reinvent the wheel”. Therefore, following a literature review, the specific requirements of KSUCTA will be investigated. From these requirements a concept for the “Design of a low-cost client/server GIS” will be documented, which will then be used as a guideline for its implementation. Once the requirements and the concepts are clear, work can begin on selecting component technology and building the prototype.

Once the prototype is up and running, it will be necessary to test the system thoroughly with many different loading scenarios to assess its capabilities. This testing process focuses on issues like the number of concurrent users possible, the amount of data transfers possible, the back-up and restore capabilities etc. with all of these being tested using different network loadings.

Such systems are not new, but many are commercial solutions that require big budgets, or else they are quite narrow in their focus and therefore not transferable from the specific situations for which they were developed. The goal is to develop a system that, while tailored to KSUCTA’s specific requirements, might be transferable and and able to be customised for other situations, such as other universities wishing to implement such a *low-cost* client/server GIS.

1.4 Expected results

The results will include the following three points

1. The specification for a low cost client/server geographic information. This will be at quite a conceptual or high level but will deal with standards, and with platform and technology issues.
2. Knowledge of the requirements of an institution that can be met by such a system. It is these requirements that will detail such issues as the amount of data involved, the loading on the system (i.e. the number of concurrent users) and bandwidth requirements. This will also include any limitations that there may be in comparison to a commercial solution.
3. Building a prototype. This will be a working model that is able to demonstrate

the full capabilities of the system to fulfil the requirements.

1.5 Constraints and risks

As mentioned above KSUCTA is really just starting out in GIS and is, therefore, not carrying the “proprietary baggage” of an organisation that has been using GIS for 15-20 years. That is, there is not a huge volume of data in a format that can be read only by one application and there has not been a huge investment in a particular software platform that can understand only one or two data formats. As we will be creating a completely new system it is desirable, right from the beginning, to adhere to Open Geospatial Consortium (OGC) standards for data interchange and to select components that adhere to these standards, as OGC compatibility will enable far simpler extension and interconnection with other systems. However, while the adherence to OGC standards could greatly simplify future development and expansion, its practical implementation will need to be fully investigated to ascertain the feasibility of going down this path.

It will also be desirable to make use of open source software where it is both available and suitable to the situation. This does not preclude using commercial software, but the emphasis will be on low cost.

Finally, the data providers/owners will need to be made aware of the project, as some of the data that will be used does not belong to KSUCTA and is not necessarily in the public domain. This will be of particular importance if the map server is made available to the general public over the internet.

1.6 Issues that will not be discussed here

This thesis discusses Open GIS and touches on its technical implementation but, for Open GIS to work, there needs also to be a political commitment. This means that those who control the data have to be willing to see it as a public resource of great value to the country as a whole, when it is made available to the whole country. It is, however, not the aim of this study to explore this issue in the Kyrgyz context, as it really needs to be resolved at a senior government level.

Also, while discussed briefly in chapter 2 Literature, it is also not the purpose of this study to implement an INSPIRE-like infrastructure in Kyrgyzstan or Central Asia. However, the principles of the INSPIRE initiative [INSPIRE] provide an excellent guide when building databases, especially where they relate to cataloguing and attaching metadata.

1.7 Intended audience

1.7.1 Diction

The intended audience of this thesis are Geographic Information (GI) professionals, as opposed to Information Technology (IT) professionals. Therefore, the language used assumes the reader is familiar with GI science, spatial data, map projections, geographic analysis, etc. It will however, seek to clearly explain and clarify IT terms that for many GI professionals are considered "black box" issues only required by IT professionals.

1.7.2 Professional and vocational profoundness

Although GI science is rooted in IT and has been since its computerisation in the 1960s, it has grown into a huge discipline with many sub-disciplines and it is impossible for one person to maintain an in depth knowledge of all there is in GIS and the IT that supports it. The aim of this thesis, therefore, is to help the GI professional, particularly those working alone or in small teams with little or no IT support, to branch easily into the world of on-line map serving applications.

1.8 Thesis structure

The thesis contains seven chapters plus a sizeable amount of material in the appendices – it is structured as follows:-

- Chapter 1 Introduction - sets the scene, providing background to this study and highlighting issues that are particularly significant to the situation in Kyrgyzstan in general and KSUCTA in particular. Also listed in the introduction are the expected result areas and the methodology that will be followed.

- Chapter 2 Literature - lists works particularly relevant to Open GIS and Open Source issues, and shows how technology advances in recent years have paved the way to the rapid development in these two areas.
- Chapter 3 Approach – Covers the theoretical foundation of Open Source and Open GIS, and also outlines the methods applied.
- Chapter 4 Project Description - divides into the concepts for developing a client/server GIS and its implementation, where the implementation describes the test area, the two levels of test data and the tools used. The section on tools is reasonably sizeable and bears witness to the considerable depth and maturity of open source software available.
- Chapter 5 Results - refers back to the three result areas of the specification, the requirements and the building of the prototype listed in section 1.4 above. This chapter details issues such as data quantity and concurrent access requirements, as well as the whole testing process.
- Chapter 6 Analysis of Results - looks deeper into the results and and digs into some of the issues arising from them. Of particular importance are those that deal with data security and robustness of the system. The *issues arising* are divided into three sections covering PostgreSQL/PostGIS, UMN MapServer, and the data itself. Finally, this chapter also deals with the the implementation of OGC specifications.
- Chapter 7 Summary, Discussion and Future Work - brings the whole study together and looks at just what has been achieved and what needs to be done for the system to become a production solution. Also covered here are additional advancements that could be done, or should at least be looked at in the future.
- The appendices - Appendix A contains a brief description of the OGC services used in this study. Appendices B to E contain much of the implementation detail

for the various components chosen, While Appendix F contains an interesting table comparing the functionality of various Open Source GIS software, of particular importance here is their adherence to OGC standards.

CHAPTER 2 LITERATURE

Whilst there is a huge wealth of literature available that could relate to this topic, much of it is highly technical and relates to the nuts and bolts implementation. There is also a significant volume of literature that deals with organisational issues at an overview, but much of this is quite high level with a considerably smaller amount dealing with organisational issues at an operational level. This chapter reviews some of the more recent relevant literature, broadly dividing it into three topic areas of *Open Source GIS*, leading into *Open Geospatial Consortium* issues, and finally touching on *Spatial Data Infrastructure*.

T Mitchell discusses "*the coming out of the OGC community.*" He is referring to the "Open Source GIS / MapServer User Conference Ottawa (June 9-11 2004)" where the audience, amongst many others, included GIS analysts and web developers. He goes on to say "*The fact that these two realms are overlapping regularly is exciting in itself*" and further *"Often the main benefit of open source GIS and mapping tools appears to be the low cost, especially when compared to the commercial alternatives. However, the greater strength, we are realizing is the vibrant community support and also the power of technology that is ahead of the commercial software curve in many respects.*" [Mitchell 2004]. Although this article was written some years ago it is still highly relevant, as GIS continues to grow into the capabilities provided by the ongoing development of web mapping sties like Google Earth and Microsoft's Mappoint. The comments posted to it provide some very good background information on the MapServer project and a number of useful links and pointers of where to look for more details of open source GIS. This article is relevant to this study because it focuses on how the smaller players in the GIS industry are also able to enter the web mapping arena.

Billed as "*an open source development environment for building spatially-enabled internet applications*" the MapServer project originally developed by the University of Minnesota (UMN) in cooperation with the National Aeronautics and Space

Administration (NASA) and the Minnesota Department of Natural Resources (MNDNR) offers almost a ready built solution for internet mapping. However, it never aspires to be a fully featured GIS but rather "*excels at rendering spatial data (maps, images, and vector data) for the web.*" [UMN MS 2007]. There is a large community and plenty of 'getting started' and advanced helps available for those wanting to utilise this. There are also a number of pre-packaged implementations of MapServer available, including "HostGIS Linux", (billed as "*Everything you need to start serving web-based maps in minutes*") where the HostGIS company also offers a web hosting service [Perry/ Mosheh 2005], and Maptools (or MS4W) for the Windows platform [DMS 2007].

"Open Source and Interoperability" is a presentation by Geoff Zeiss, Director of Technology, Infrastructure Solutions Division, Autodesk, Inc. to the Map Africa 2006. This is a bit of a media push for OGC but within it the author talks of creating an "*IT ecosystem in Africa*" and says that open source is important as it offers a "*Low cost barrier to entry*" [Zeiss 2006]. This is as important in Africa as it is in Kyrgyzstan

PostGIS claims to provide a very desirable open source answer to a commercial spatial database like Oracle Spatial and also complies with Open GIS specifications. On its website it makes the following claims: *PostGIS adds support for geographic objects to the PostgreSQL object-relational database. In effect, PostGIS "spatially enables" the PostgreSQL server, allowing it to be used as a back end spatial database for geographic information systems (GIS), much like ESRI's SDE or Oracle's Spatial extension. PostGIS follows the OpenGIS "Simple Features Specification for SQL" and has been certified as compliant with the "Types and Functions" profile.* [Refractions 2005].

In 2004 John Simpson published a very useful introduction to GML for the website XML.com. In it he states "*One primary motivation for developing GML in the first place was to simplify delivery of maps over the web"* and goes on to say *"developers seem to be focusing (for now) on using GML as a common data interchange vehicle.*" [Simpson 2004]. He points out that a GML document is not so much a document like a PDF file that can be opened in an Acrobat Reader® type application; rather it is an XML based schema, modelling technique that allows for the building of application schemas and the reliable exchange of spatial data.

The Infrastructure for Spatial Information in Europe (INSPIRE) has produced a document, of which the ultimate goal is to achieve/create "*an open, cooperative infrastructure for accessing and distributing information products and services on-line*." [Smits 2002]. This document, while being sizeable, contains many high level principles like access and usage conditions for spatial data, and considerable detail for their implementation like which meta data fields should be included. The document is of value as it contains the result of considerable technical research into many of the issues that are very relevant in the Kyrgyz/Central Asian environment, or that will be relevant once the political implications of these issues are worked through.

In a report on geospatial interoperability NASA states the following "*Geospatial Interoperability is the ability for two different software systems to interact with geospatial information. Interoperability between heterogeneous computer systems is essential to providing geospatial data, maps, cartographic and decision support services, and analytical functions. Geospatial interoperability is dependent on voluntary, consensus-based standards... These geospatial standards are essential to advancing data access and collaborations in e-Government, natural hazards, weather and climate, exploration, and global earth observation.*" [NASA ROI 2005]. This report highlights the importance of geospatial standards while at the same time stating their voluntary and consensus based nature. In the modern world, where organisations are constantly aware of time pressures and the financial bottom line, there has to be good reason for them to take on something optional that is going to involve a significant time investment. As stated in section 1.6 it is not the intention of this study to implement such standards, but it is hoped that the systems identified here allow for geospatial interoperability by implementing some of the main OGC standards.

Geospatial interoperability is particularly important in cross-border issues, as many of the national boundaries in Central Asia are somewhat arbitrary in nature. Nazarkulova (2007) proposes an SDI architecture for a cross-border national park which uses a "*distributed node network*". The example given is for a national park bordering Kyrgyzstan and Tajikistan, where data is stored and maintained separately and independently on both sides of the border, while being made available through a common SDI agreement. [Nazarkulova 2007].

This study does not seek to repeat any of the work mentioned above but rather to use it, along with additional literature, and then apply it to the building of a specific system for a specific environment. Having achieved this it will then document the implementation process and highlight the issues encountered.

CHAPTER 3 APPROACH TO DESIGNING A LOW-COST CLIENT/SERVER GIS

Stated in the introduction in section 1.3 was the desire to take a *top down* or *big picture* approach, knowing that in KSUCTA's case there are few existing constraints that need to be taken into account. This chapter discusses the theory and methods involved in considering such an approach in the KSUCTA situation.

3.1 Theoretical Foundation

It would be fairly easy to build the type of system indicated in Figure 1 on page 3 using "off the shelf", commercially available products. However, the aim of this study is to investigate a *low cost* solution, particularly one which uses *open source* applications. Additionally, it is not desirable to be locked into a single vendor for all components, but rather to be free to choose components from various vendors, so as to be able to take advantage of the full breath of available technology. It is, therefore, necessary that a suitable data format be chosen to which all components should adhere. For this reason Open GIS standards, like the Web Feature Service (WFS) and the Web Coverage Service (WCF), are desirable and the implications of their implementation need to be fully considered. These two issues will now be briefly discussed.

3.1.1 Open Source

From the website of the Open Source Initiative is the following opening statement: "*Open source is a development method for software that harnesses the power of distributed peer review and transparency of process. The promise of open source is better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in.*"[2] [OSI 2007]. Or, as Wikipedia simply states a "s*et of principles and practices on how to write software.*" [Wikipedia OS 2008].

2 The last phrase of this quote is very strong language and reflects the author's strong feelings on the matter.

It is unfortunate that to many people open source simply means free software, when it is, in fact, so much more. Open Source is about a continuous development cycle for software, that uses a huge pool of human resource. It also invites all users to become part of the beta testing process and even to contribute code.

For a developing country like Kyrgyzstan with its modest software budgets, Open Source provides quality legal software supported by a large user community. In addition this software is now developed to such an extent so as to be able to offer many of the capabilities of expensive commercial software.

One of the arguments that is often used against open source software is that it is not mature, robust enough, or sufficiently supported for a production environment. But in an article that appeared on Network World in July 2005 it was shown that there is actually a move by companies towards the deployment of open source tools to support *business critical* functions. [Garretson/Fontana 2005]. Additionally, if a particular programme does not quite fulfil the requirements of an organisation then because the source is freely available, it may be customised and even modified far more than any commercially available program.

3.1.2 Open GIS

Serving geographic data from a database to only pre-designed web pages and predefined applications will generally simplify initial design, but can often create more work in the long term. An example of this would be where an additional application is required by someone, either not connected with the original project or from a completely different organisation, wanting to access the data on-line. A lot of extra work and research may be needed to discover the structure and capabilities of the database. Additionally, as it was intended that the system support both a bidirectional work flow, and the sharing of the actual data, it was desirable that these features be implemented using OGC specifications.

Therefore, the three OGC specifications of WMS, WFS and WCS were considered appropriate for this study. A brief introduction to these services is included in Appendix A on page 61.

- The Web Mapping Service (WMS) is used for the creation of a static map, usually in the form of a ".jpg" or ".png" image file, of the kind used in many internet mapping applications. With WMS, intelligent data is not transferred but, rather only a map representation of the data.
- The Web Feature Service (WFS) provides for the retrieval and updating of the actual data. This data transfer is achieved using GML and can perform standard editing operations of creating, deleting, updating on a feature by feature basis as well as feature locking and querying.
- The Web Coverage Service (WCS) supports the electronic retrieval of data as coverages, or whole layers, as opposed to working on a feature by feature basis.

All these services support the standard request "GetCapabilities" to return information about the data available; responses to this request are delivered in XML2 format and enable the user to issue further requests to query the database, create maps, and even download the data itself.

OGC standards also define a "*simple feature geometry*" and the document describing this states the following: "*The purpose of this specification is to define a standard SQL schema that supports storage, retrieval, query and update of simple geospatial feature collections via the ODBC API. A simple feature is defined by the OpenGIS Abstract specification to have both spatial and non-spatial attributes. Spatial attributes are geometry valued, and simple features are based on 2D geometry with linear interpolation between vertices.*" [OGC SF 1999]

Desirable as it seemed to adhere to this standard, it has proved somewhat restrictive in that, for example, it does not allow for 3D (or 4D) geometries, nor does it allow for the inclusion of map projection information. It may be, therefore, necessary to use a superset of the simple feature geometry for access within KSUCTA. However, any connection to the data server that uses the internet should be completely OGC compliant.

3.2 Methods Applied

The methodology used in this study involved taking open source software and building an OGC compliant client/server GIS involving the three clearly distinct components of a back end database server, a central data server and one or more client applications as indicated in Figure 1 on page 3. In reality however, the application that serves the data to the client applications is usually part of the DBMS. For example, the Oracle DBMS serves data from an Oracle Spatial database, or the PostgreSQL DBMS serves data from a PostGIS database. Figure 2 below indicates how this operates. Note that the actual choice of DBMS and client applications is discussed in section 4.2.3 Tools, beginning on page 27.

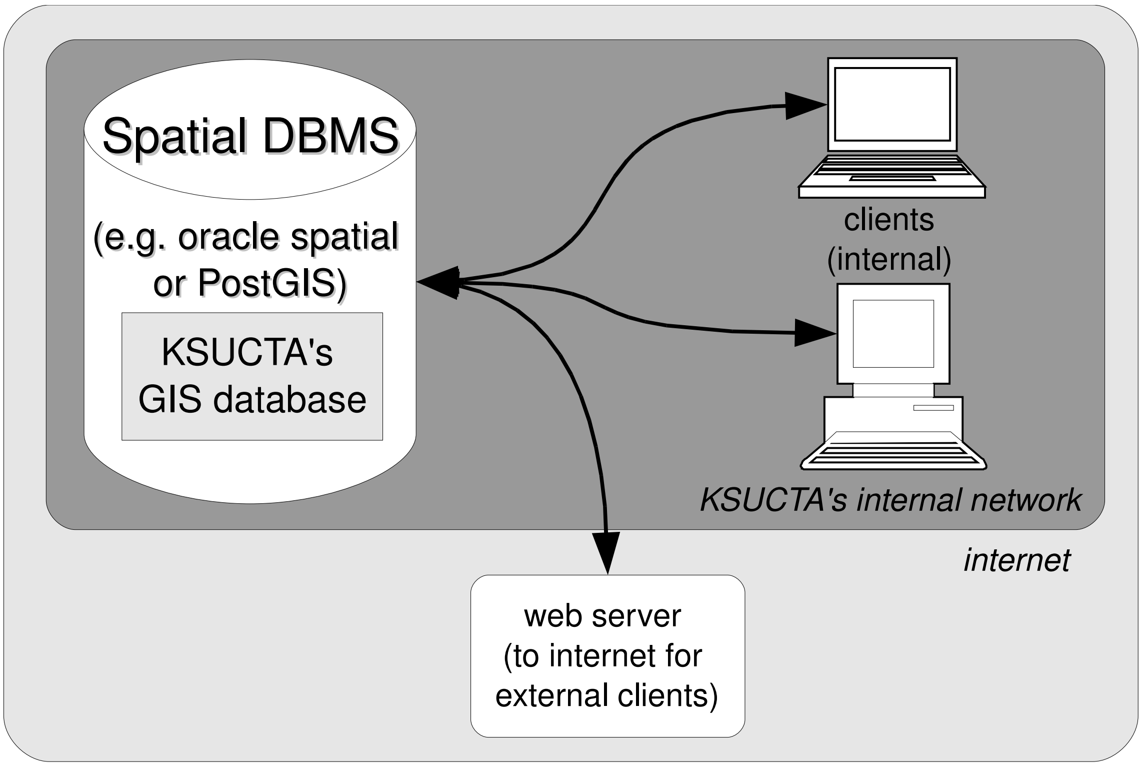

Spatial DBMS
(e.g. oracle spatial or PostGIS)
KSUCTA's GIS database
clients (internal)
KSUCTA's internal network
internet
web server (to internet for external clients)

Figure 2: Showing an application of how the map server could be implemented.

Testing and tuning is of utmost importance for any information system, and in an environment such, as a developing country, there are some unique challenges, with a less than stable power supply and network stability problems. It is vital that the components chosen are robust and that the system as a whole be tested, not just for its ability to cope with the required loadings but also for its ability to recover from power outages and network crashes. It is also important to know how easy it is to restore a working system in the event of a complete loss of the server computer. These issues are discussed more fully in section 6.1 starting on page 45 below.

CHAPTER 4 PROJECT DESCRIPTION

The first part of this chapter deals with the concepts for developing a low cost client/server GIS, along with listing some specific requirements unique to the Kyrgyz environment. The second part gives a detailed treatment of the implementation, which includes discussion on the type and volume of data to be included in the trial and the tools that will be used.

4.1 Concept for developing client/server GIS

It is important to view each component, the database, the data server, and the clients, as part of the geographic information system and not the system itself. That is, the GIS is not the data server, nor is it a particular client application. Also it is undesirable that the data server be required to serve numerous data formats, or that all of the clients be required to read a proprietary data format in order to access the data. For this reason it is desirable to store all data in a Spatial Database Management System, (SDBMS) which the data server is able to read and then to output to a generic, or OGC, format, which can be understood by a variety of applications.

There is nothing new in this sort of approach, but at KSUCTA there are some unique conditions that need to be addressed:

1. **Cost** – Kyrgyzstan has an emerging economy and is financially limited compared to more developed countries; therefore, if any solution were to be sustainable in the long term it would need to pay attention to the issue of financial cost. While many commercial software providers are very generous in making their products available to universities at low cost, it was not considered practical to base the KSUCTA system on these, as software agreements usually do not allow use for commercial purposes nor for use outside the university environment.

2. **No existing SDI** – Kyrgyzstan currently has no existing Spatial Data Infrastructure (SDI), although moves are being made to coordinate state agencies

in this direction. [Karypov/Kazkanova 2008] and [Jantaev/Zubovich 2008]. Dialogue with these agencies, especially the State Land Registry Office was, and will be in the future, important to ensure compatibility with, and to even influence, the shape of any national spatial data standards.

3. **Russian Language** – While English is considered by many the *de facto* standard language for the IT world, it is in fact not widely spoken in Kyrgyzstan. Therefore, any client programs chosen and applications developed need to consider the following two capabilities.

   - Russian and Kyrgyz text, using Cyrillic font will be ***required*** in any database fields containing local place/street/owner names etc.
   - Russian and Kyrgyz text, using Cyrillic font ***should*** be available as an option on all client interfaces.

4. **New technology** – Over recent years a considerable amount of new technologies, such as low cost location enabled cellular phones and powerful hand held data capture devices, have been developed. Much of this new technology has been and will be in the future a great benefit to this project. This development is set to continue and it is, therefore, important that KSUCTA's system be robust and flexible enough to take advantage of these developments without needing to undergo a major redesign.

4.2 Implementation

This implementation section describes the process of the selection of the tools that were used in the project, as well as the choice of data that were used. The specific details of quantities of data and more detailed tuning of software components follows in chapter 5 Results.

4.2.1 Test area

The test area was the KSUCTA Maldybaev Street campus. All database and web serving software were installed on a test server, a dual core Pentium 4 computer running a licensed version of Windows Server 2003 R2, located in the Department of Geodesy and Geoinformatics. All software used in this study is either fully licensed for education use or is Open Source.

4.2.2 Test datasets

One of the problems of using GIS in Kyrgyzstan is the availability of quality GIS data. Therefore, for the purpose of this study, two levels, or scales, of data were selected as being representative. These are an overview level for small scale mapping and a detail level for larger scale mapping. For the initial selecting and testing of software, as detailed in section 4.2.3 Tools, only the overview level was used.

Overview level data

The overview level data are suitable only for small scale and overview mapping. Figure 3 shows a visual representation of them.

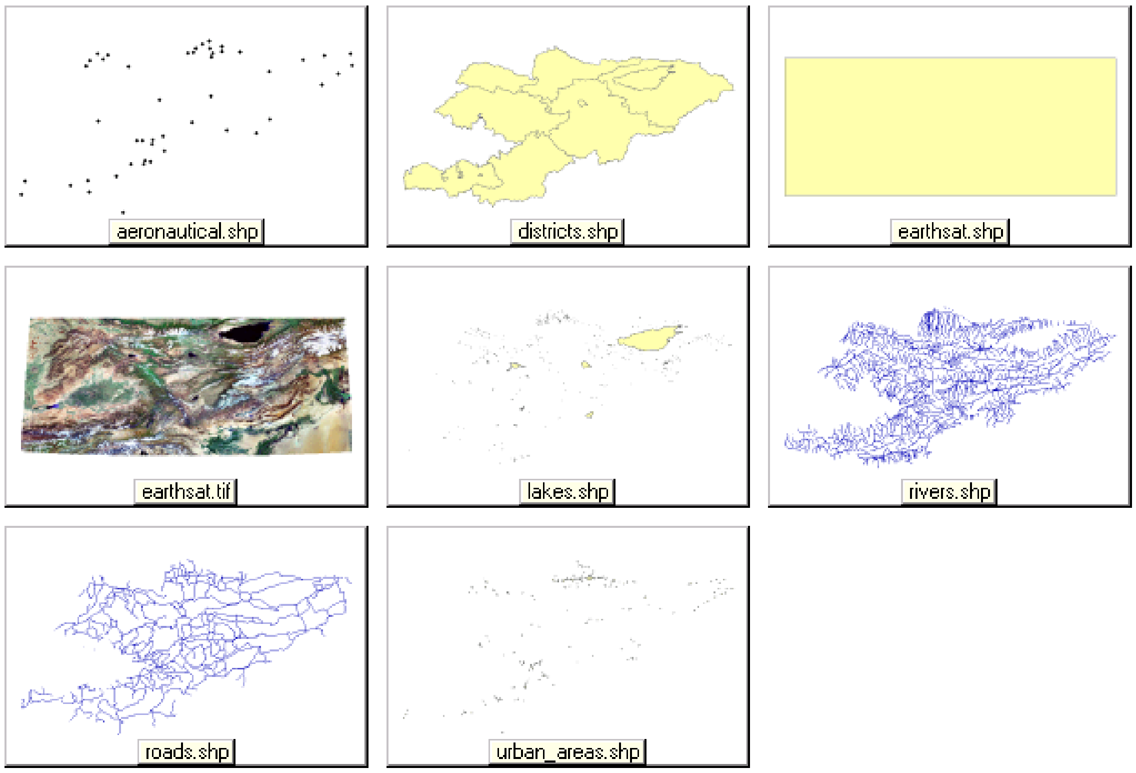


Figure 3: The layers in the test dataset – overview level.

- The *aeronautical* layer from the Digital Chart of the World (DCW) [ESRI/DMA 1993] provides a point layer with attributes.
- The *districts* layer, captured at KSUCTA, and the lakes and urban_areas layers from the DCW provide polygon layers with attributes.
- The *roads* and *rivers* layers from the DCW provide line layers with and without attributes respectively.
- Finally the raster layer *earthsat* is a clip from the ESRI supplied "Asia west 150m Earthsat" image [Earthsat 2004]. This was included to see how raster data performs in the system. It should be noted that a requirement of MapServer is that a tile layer be provided for raster images; this is what the shape file earthsat.shp represents.

Detail level data

The following data are suitable for larger scale mapping and consist primarily of selected areas of 1:50000 topographic vector data and scanned maps, colour composite Landsat TM images, and NASA's Shuttle Radar Topographic Mission (SRTM) digital elevation data.

1:50000 topographic data

Four 1:50000 map sheets, covering an area around Bishkek, were scanned and vectorised by the United States Geological Survey (USGS) for the Kyrgyz State Institute of Seismology. Figure 4 below shows the extent of these data. In addition to the data listed in Table 1, the TIFF images of the scanned map sheets were also supplied, along with a single IKONOS scene,[3] in both colour infra-red and natural colour.

3 This image had a one metre resolution which meant that it was actually a panchromatic–sharpened colour image as the raw IKONOS colour image has a spatial resolution of four metres.

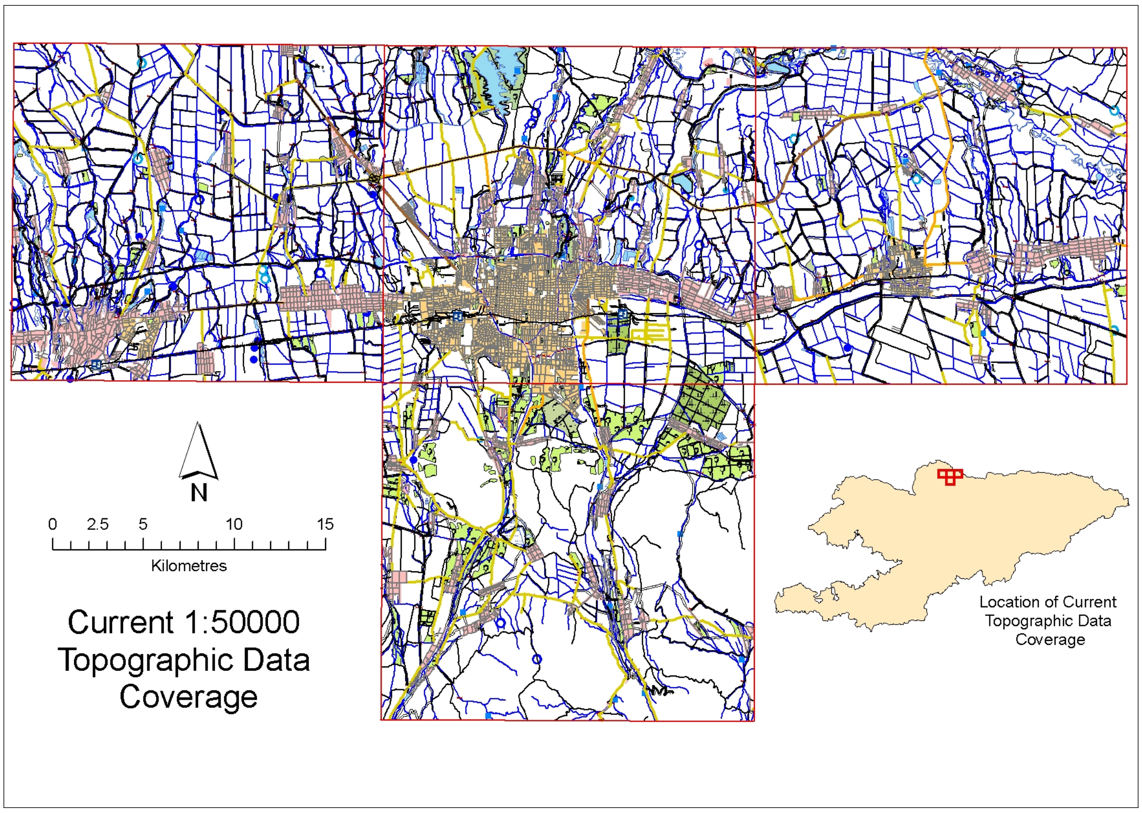

N
0 2.5 5 10 15
Kilometres
Current 1:50000 Topographic Data Coverage
Location of Current Topographic Data Coverage

Figure 4: Map showing the extent of 1:50000 topographic data around Bishkek.

Table 1: A list of 1:50000 data sets used in this project.

Layer name/description	number of features
Factories, mines, quarries, power-stations, cultural landmarks	406 points
Marks of vegetation description	10 points
- Transport routes	
- Transportation buildings	77 lines, 5 points
- Railroads	146 lines
- Roads	12387 lines
- Bridge	14 lines, 643 points
National border	6 lines
Pipelines and lines of electrical transmission	1298 lines
City blocks	5151 polygons
Hydro	
- Mark of water elevation	29 points
- Water wells and springs	46 points
- Rivers, canals, and drainage ditches	3850 lines

Layer name/description	number of features
- Marshes and swamps	107 polygons
- Rivers, lakes, and reservoirs	424 polygons
Woods/forests	368 polygons
Relief	
- Marks of height above sea level	439 points
- Cliffs, scour, and dry riverbeds	421 lines
- Contour lines	4528 lines
Populated places and separate buildings	
- Small Populated Places or separate buildings (for desert regions)	4610 points
- Single building without landmark significance	2931 lines
- Separate buildings	482 polygons
- Populated Places	170 polygons
Quarry, industrial enterprise, and cemeteries	27 line, 324 polygons
Rock/rocky cliff	60 polygons
Soil	47 polygons
Dam, mole, water-pipe, and cliff	829 lines

Landsat TM data

There are 20 Landsat themes that cover the territory of Kyrgyzstan for which KSUCTA possesses all TM bands as TIFF files. In addition, KSUCTA also has the larger colour composite themes for the whole country (see Figure 5 below). These, however, are very large and currently stored as MRSID images, each about 200MB, whilst their uncompressed size is 4.7GB each.

For the purpose of this study it was decided to serve only the colour composite images, although, in principle, there is no reason why all images could not be served on the system.

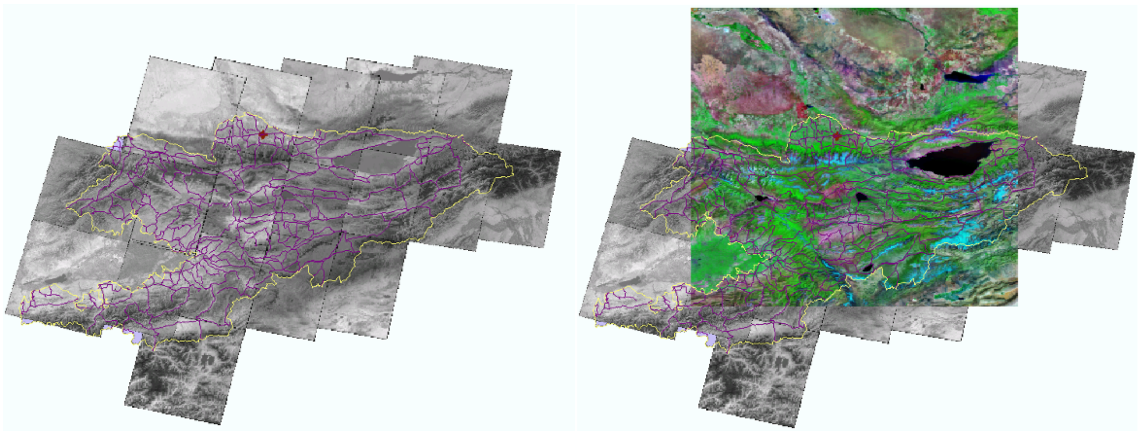


Figure 5: Map showing the standard Landsat TM theme coverage (left) c.f. the larger colour composite themes (right)

SRTM digital elevation data.

NASA’s 90m digital elevation data, resulting from the Shuttle Radar Topographic Mission (SRTM), originally contained a number of “holes” i.e. areas that, due to cloud cover and shadow effect, resulted in no data values. This dataset has recently been upgraded by applying a hole-filling algorithm [CGIAR-CSI] and using auxiliary DEMs. As a result, what was a very good dataset has now become an excellent dataset and a very valuable resource. It was decided to include it in the detail level dataset, as it forms a very good background image for general mapping and is highly useful for obtaining an elevation quickly from a mouse click. There are six tiles, each one 5^o x 5^o square or about 425km x 425km, covering the territory of Kyrgyzstan. It was decided that for each of the images one version of the raw image should be served and one version which had been processed into a relief shaded hypsometric image as shown in Figure 6.

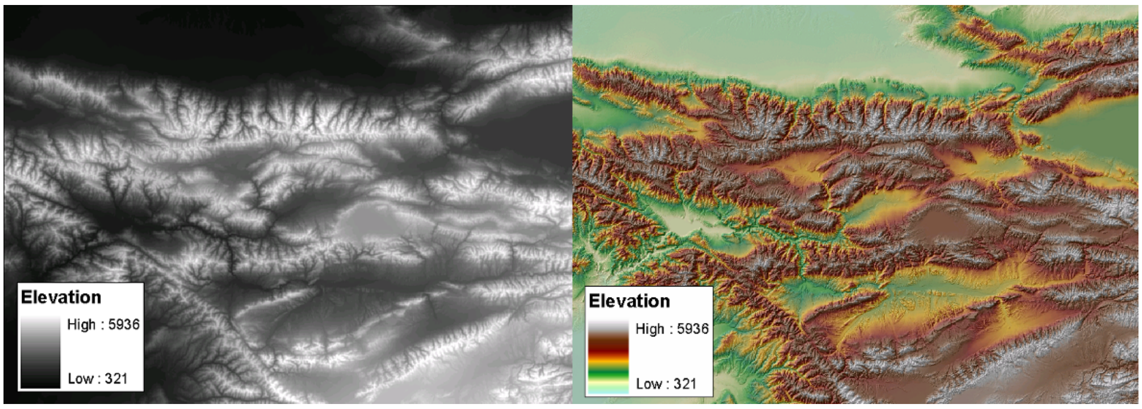


Figure 6: Map showing the raw SRTM image (left) c.f. SRTM image showing relief and hypsometric shading (right).

Map projections and coordinate systems

For the initial part of the study, that is the selecting and testing of software, as detailed in section 4.2.3 Tools, all data was converted to the “UTM Zone 43 North” map projection. However it was desirable not to have to pre-process the data in anyway, it was therefore deemed best to store all data in its original projection and re–project on the fly. Table 5 below compares the Pulkovo 1942 datum, used for much of the existing mapping and data in Kyrgyzstan, against WGS 1984. Figure 7 shows the oblasts (regions) of the Kyrgyz Republic displayed using "Pulkovo 1942 Gauss Kruger Zone 13" map projection, with adjacent GK and UTM zones shown

Table 2: The comparison of Pulkovo 1942 and WGS 1984 datums

```
Name: GCS_Pulkovo_1942                        Name: GCS_WGS_1984
Alias:                                        Alias:
Abbreviation:                                 Abbreviation:
Remarks:                                      Remarks:
Angular Unit: Degree (0.017453292519943295)   Angular Unit: Degree (0.017453292519943299)
Prime Meridian: Greenwich (0.0000000000000)   Prime Meridian: Greenwich (0.0000000000000)
Datum: D_Pulkovo_1942                         Datum: D_WGS_1984
  Spheroid: Krasovsky_1940                      Spheroid: WGS_1984
    Semimajor Axis: 6378245.00000000000000        Semimajor Axis: 6378137.00000000000000
    Semiminor Axis: 6356863.01877304730000        Semiminor Axis: 6356752.31424517930000
    Inverse Flattening: 298.30000000000001        Inverse Flattening: 298.25722356300003
```

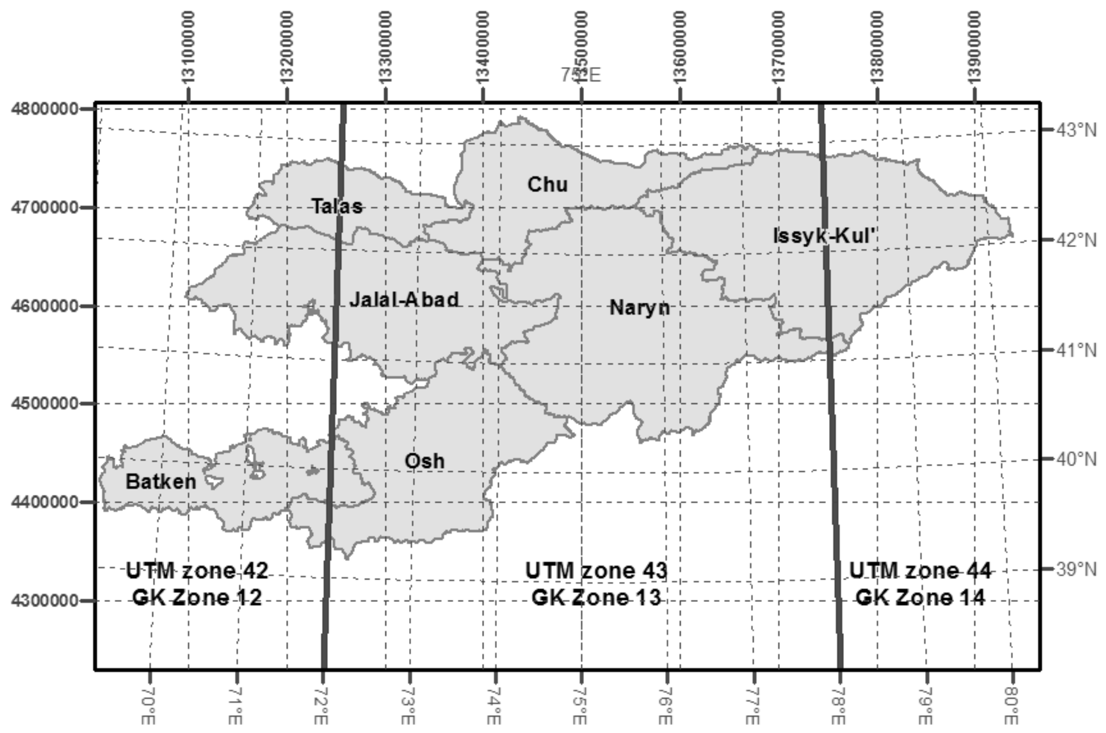


Figure 7: Map showing the oblasts of the Kyrgyz Republic displayed using "Pulkovo 1942 GK Zone 13".

4.2.3 Tools

This section describes the software tools used in this study and details their installation and basic configuration. It should be noted that this section does not cover all possible tools, but there is a fuller list of open source GIS packages, summarising key functionality relevant to this study, listed in Appendix F.

A final note in this introduction to the Tools' section relates to ArcGIS. Although not Open Source, KSUCTA does hold a 25 seat education licence to the ArcGIS Desktop suit. Because of this, and because ESRI is such an industry leader, it is necessary that any system developed seriously considers integration with the ArcGIS product range while not becoming a servant to it.

MapServer

The MapServer environment appears to offer an ideal solution to the issue of serving maps from the database. It offers a rich suite of features which include advanced cartographic output, cross platform support, scripting and development environments, and the ability to read data from a multitude of raster and vector data formats. It does, however, require a sound knowledge of GI data and an in-depth understanding of general web serving practice.

It is regarding this second requirement, that of an understanding of the nuts and bolts of web serving, that some solely GI professionals may struggle to get started with MapServer. Fortunately, there are a number of pre-packaged solutions, directed at the GI professional, to help alleviate this struggle. Two of these worth mentioning are HostGIS for Linux [available from http://www.hostgis.com/linux/download.php], which actually includes the operating system, and MapServer for Windows (MS4W) [available from http://maptools.org/ms4w/index.phtml?page=downloads.html] which has been selected for this pilot.[4]

MS4W which is available for download from MapTools.org installs pre-compiled versions of the latest MapServer, as well as a pre configured Apache Web Server, PHP, and a large selection of other utilities. There are also additional packages available for download that extend and enhance the basic installation.

An attribute of open source software, and especially a popular platform like MapServer, is that there is a large user base community and, therefore, a considerable support network available. Because of this, the new user is able to have the basic application up and running in a very short time. Additionally, any start up problems will usually have been previously encountered by others and will be recorded in the relevant FAQ listings and discussion lists.

Installation of the MS4W (MapServer for Windows) solution was very straightforward. A single downloaded file needed to be unzipped to a root directory of a drive of choice, thus creating a new folder called “ms4w”. This folder contains everything needed for MapServer to run. The second and final step was to install Apache by executing a “.bat” file, either by double–clicking it or, alternatively, by running it from a command prompt. However, in order for MapServer to produce even

4 Note that towards the end of this project HostGIS, running on Linux Slackware, was trialled at KSUCTA and performed very well.

the most basic results, as in Figure 8 below, a considerable amount more effort was required.

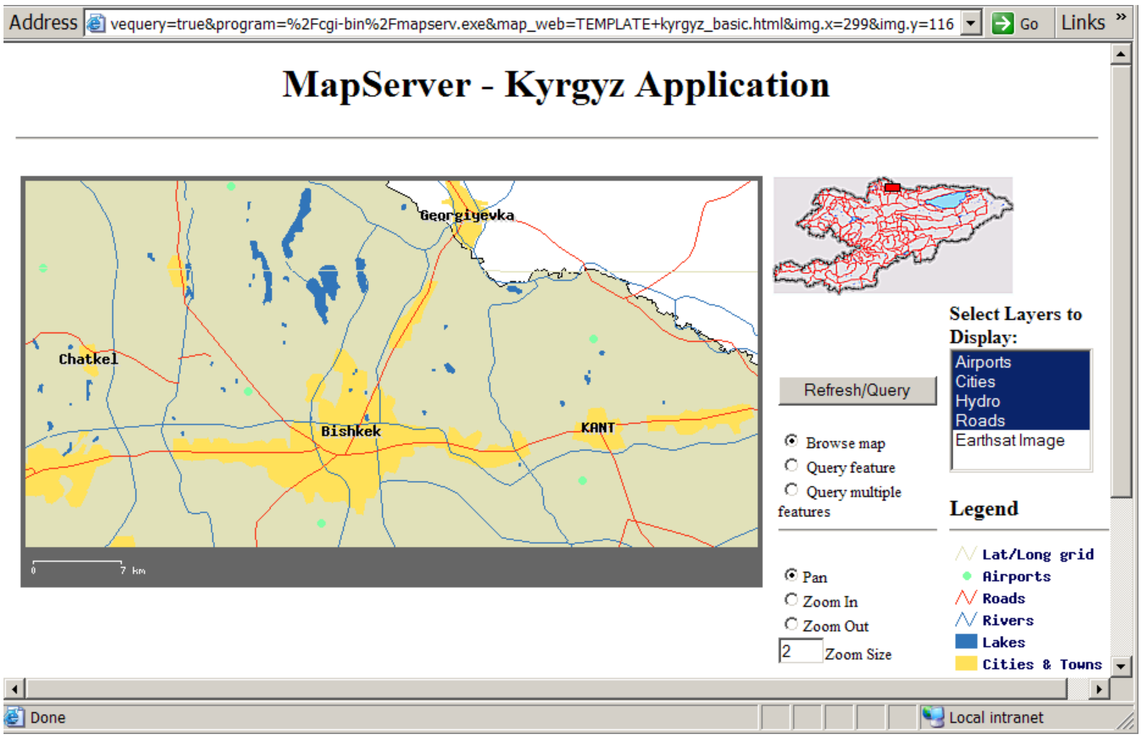


Figure 8: A screen dump of basic MapServer application using Kyrgyz data.

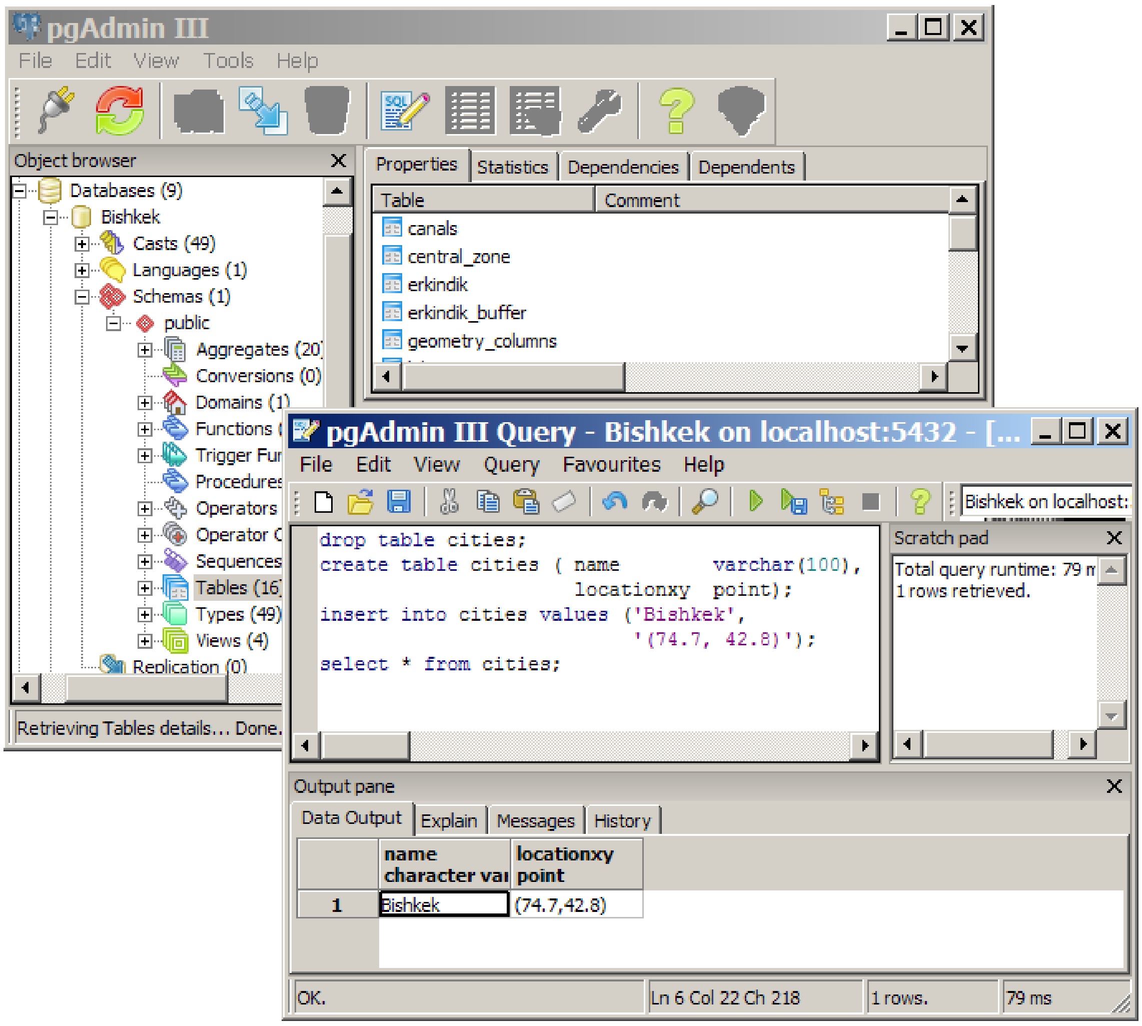
The suggested way to get started is to download the demonstration of Itasca County in Minnesota and, once it is running, to modify it step by step to point at your own data. Herein was the first delay, as version 5 of MapServer had some significant changes from previous versions and the demonstration recommended did not work. It was not, initially, obvious as to what the problem was and, although it took some time to resolve it, the solution in the end was simple, i.e. download the correct version of the demonstration. It is for problems like this that the discussion lists are very useful.

In very simple terms a basic MapServer application consists of a ".map" file, which tells MapServer where the data is and how to structure and symbolise it, and ".html" files to display the data and interact with the user. Examples of these files were all supplied with the demonstration so it was, therefore, a very simple process to edit these files and customise the application. A more in depth coverage of the "nuts and bolts" of these files can be found in Appendix B, starting on page 64.

MS4W 'out of the box' provides a relatively easy to use platform for serving GI data, which is stored in the ESRI shapefile format. It is, however, desirable that data be stored in a more generic environment, ideally within a SDBMS. Therefore, the next section investigates an open source DBMS which can be spatially enabled

PostGIS and PostgreSQL

From the outset, the thought of having to install and configure a spatial database management system and have it feeding data to MapServer is somewhat daunting to someone without an IT background. However, the entire process took less than half a day and turned out to be extremely straight forward.



```
drop table cities;
create table cities ( name        varchar(100),
                      locationxy  point);
insert into cities values ('Bishkek',
                     '(74.7, 42.8)');
select * from cities;
```

```
Total query runtime: 79 m
1 rows retrieved.
```

	name character var	locationxy point
1	Bishkek	(74.7,42.8)

```
OK.    Ln 6 Col 22 Ch 218    1 rows.    79 ms
```

Figure 9: The PostgreSQL GUI administration tool and command line SQL dialogue.

The PostgreSQL 8.2 installer, a file of 26Mb, was downloaded and installed using the standard default options. This is, in itself, an impressive product, being a fully functioning relational database with much of the functionality of large commercial databases. Therefore, with a basic understanding of command line SQL, the user is able to start work quickly on building an advanced database. PostgreSQL also has an easy to use GUI administration tool, see Figure 9 above.

Additionally PostgreSQL was able not only to store and display attributes using Cyrillic type, but was also able to have table and field names defined using Cyrillic, see Figure 13 below. This considerably eases the process of building user interfaces designed for people with very little English.

PostGIS is installed on top of PostgreSQL and is effectively a spatial template that manages spatial data through the usual SQL interface. It does, however, also allow for data to be accessed through MapServer, by ArcGIS using the "Data Interoperability" extension, and by clients built with Java and C+ like Quantum GIS.

Getting data in to PostGIS was reasonably straightforward using a utility supplied with the programme, called "shp2pqsql". This utility converts a shape file to SQL statements, making the loading of data very simple. Once the data is loaded into PostGIS the final step of displaying them in MapServer required only small alterations to the "LAYER" definitions in the ".map" file. The details of how the "shp2pgsql" utility was implemented and the exact changes needed to the ".map" file are documented in Appendix C starting on page 70.

Unfortunately, it is not currently possible to store raster data within PostGIS, so for this study these raster data were stored on the file system, but were still served via MapServer. This proved a workable solution, as although some of these raster images were very large, (see sections on Landsat TM data on page 24 and Quantity of data on page 38), MapServer was able to deliver only a screen full of data at an appropriate resolution in a fraction of a second. This is significantly faster than users needing to download entire images across the network, some of which might be in excess of many hundreds of megabytes.

ArcIMS Emulator

It is possible to connect to a PostGIS database from ArcGIS directly via the “Data Interoperability” extension, which is available as an optional extra with the ArcGIS suite. It is also possible to view MapServer data in ArcGIS via the MapServer “ArcIMS Emulator”.

This is a small but very useful add on to MapServer which is downloadable from MapTools.org. It enables MapServer to emulate ArcIMS and so allow the display of MapServer layers in ArcGIS.

This utility proved to be straightforward to install and only slightly more difficult to understand how to configure and get it running. This process is detailed in Appendix D on page 73

Figures 10 and 11 below show the results of viewing MapServer data in ArcCatalog and ArcMap

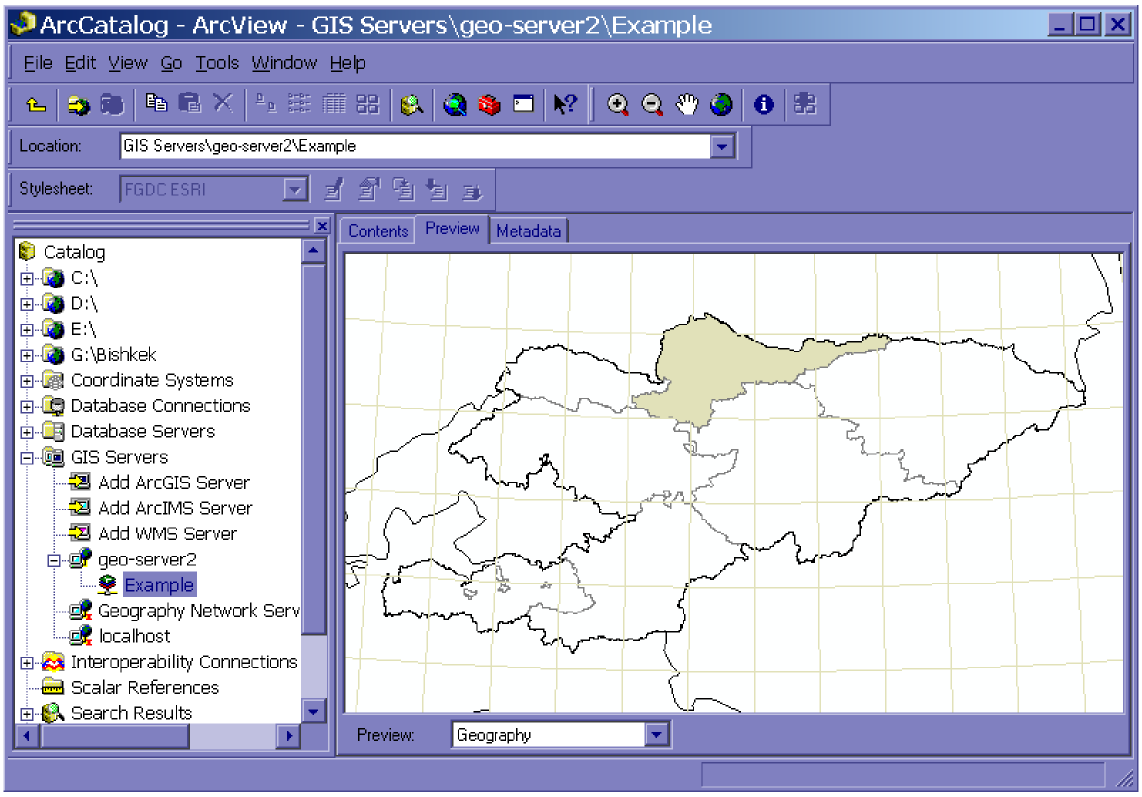


Figure 10: ArcIMS Emulator for MapServer shown in ArcCatalog.

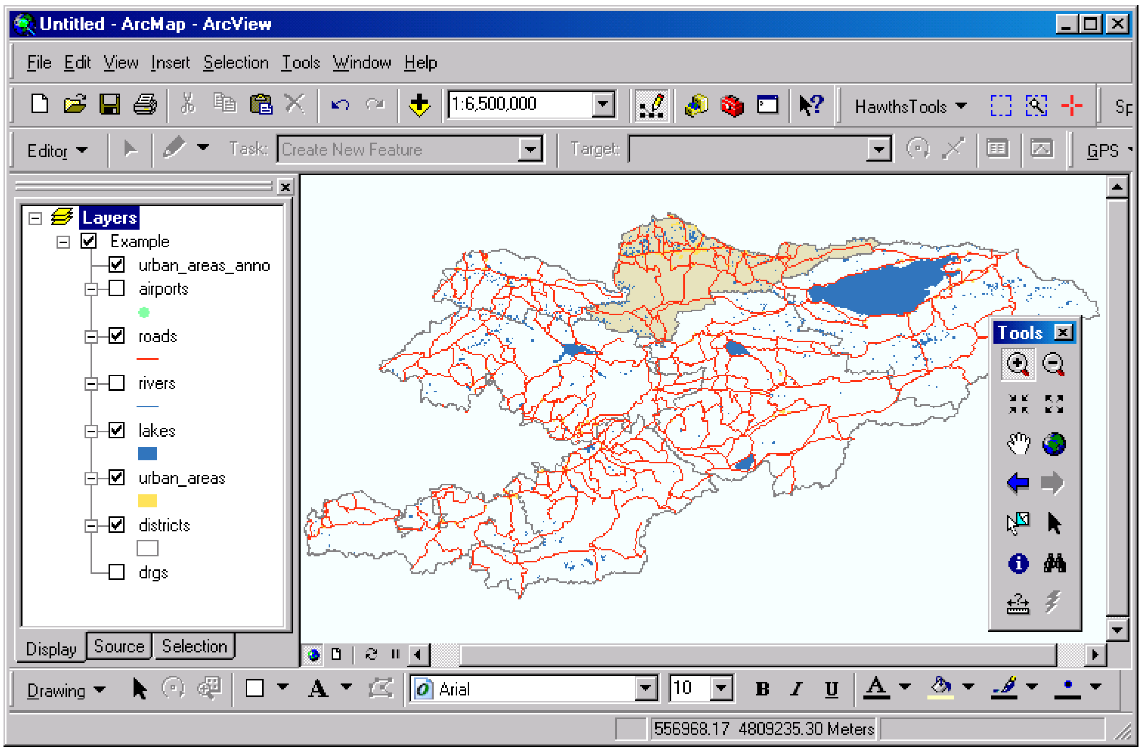


Figure 11: ArcIMS Emulator for MapServer shown in ArcMap.

Quantum GIS (QGIS)

QGIS was initially developed as a data viewer, but has evolved to quite an advanced level. It now supports an ever increasing number of raster and vector data formats with some "out of the box" analysis capabilities and the ability to add additional capabilities with a "plug-in" architecture. "*QGIS is developed using the Qt tool kit (http://www.trolltech.com) and C++. This means that QGIS feels snappy to use and has a pleasing, easy to use graphical user interface.*" [Sherman et al. 2006]

For the purpose of this study, QGIS offers a lightweight client that was able to read and write directly to PostGIS and to ESRI shape files. This included the ability to import shape files directly to PostGIS, and use the full suite of functions within PostGIS; in effect, to use PostGIS as a geoprocessing server. Additionally, using Python, QGIS was able to write MapServer ".map" files using all the settings and symbology that were used to display the data in QGIS, which greatly sped up the creation of this file.

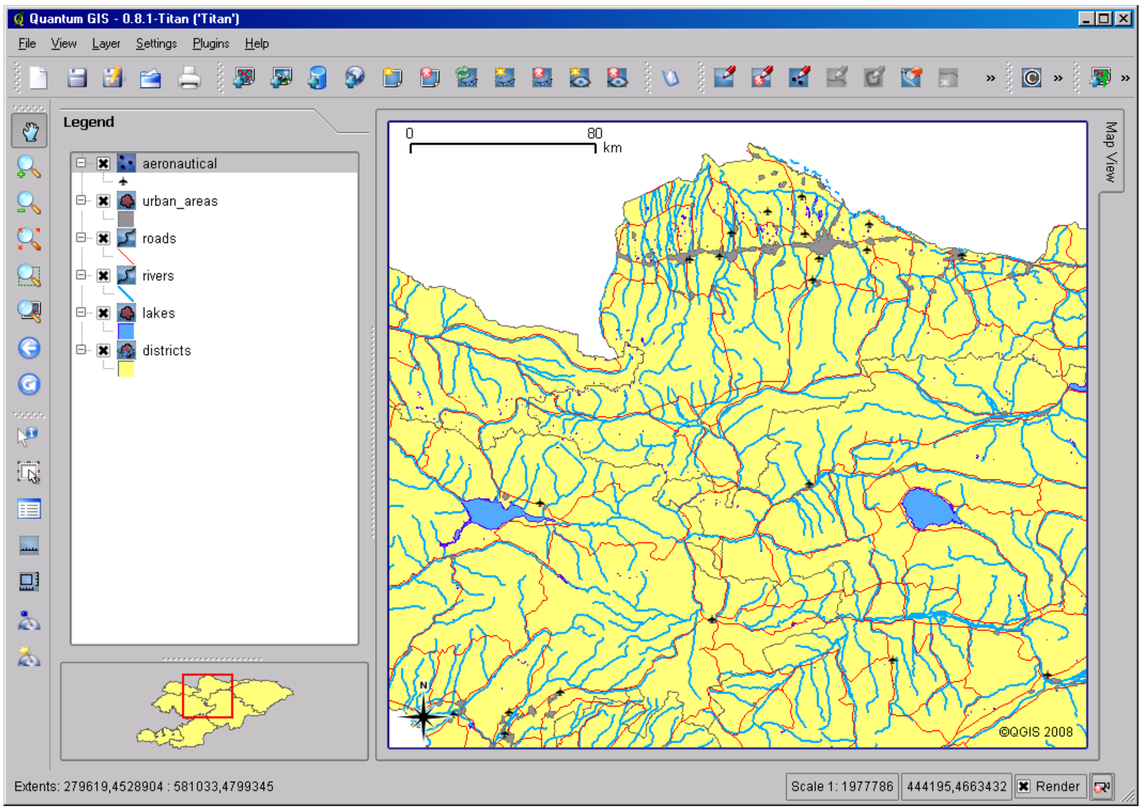


Figure 12: Quantum GIS (QGIS) displaying the test data directly from PostGIS

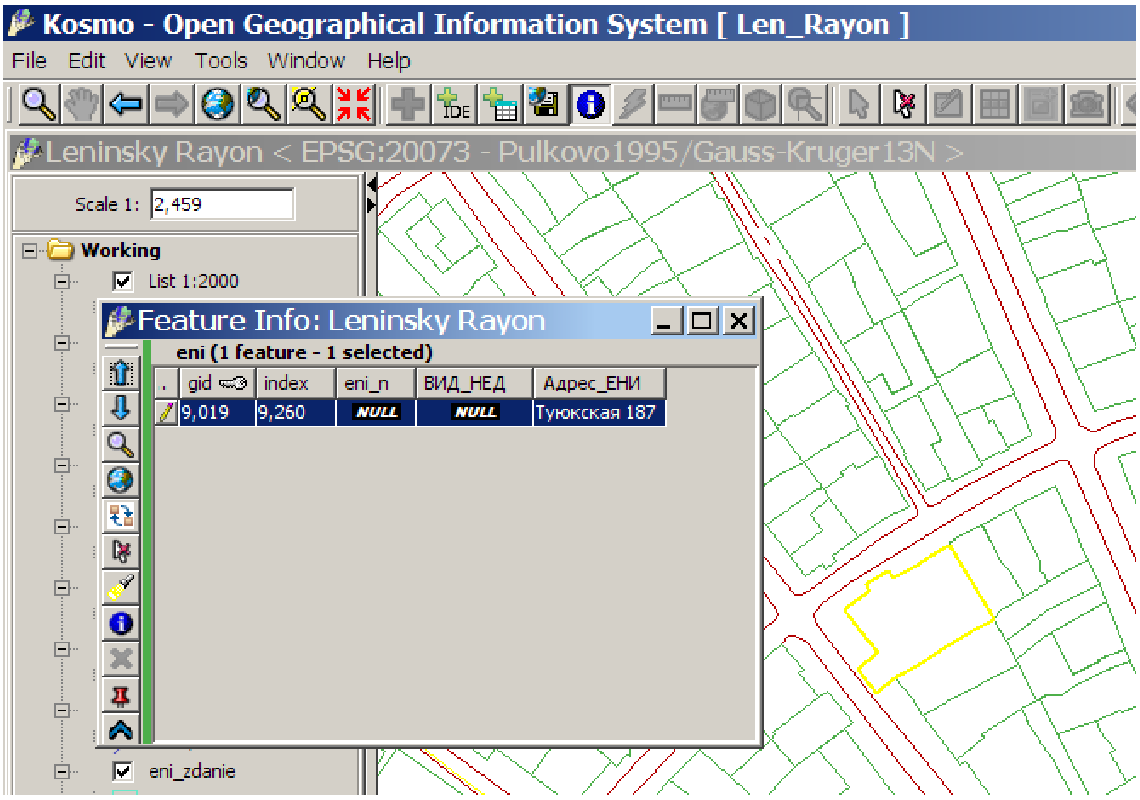
The download file of QGIS was not large and was easily installed and, as shown in Figure 12 above, was able to quickly display the test data by reading it directly from PostGIS.

Java clients

In addition to QGIS there were two more programmes used as clients for the project. Unlike QGIS, however, these were written in Java and while they ran a little slower than QGIS they offered more "out of the box" analysis functionality:

- Kosmo – Open Geographical Information System - produced by the SAIG S.L. development team in Spain. http://www.siag.es, (see Figure 13)
- and gvSIG produced by Generalitat Valenciana, IVER T.I, also from Spain.

These two were chosen from amongst many others, as they were able to access data in PostSQL and capable of displaying text using Cyrillic type, both on the map and in attribute queries and Kosmo even had a Russian language interface.


Kosmo - Open Geographical Information System [Len_Rayon]
File Edit View Tools Window Help
Leninsky Rayon < EPSG:20073 - Pulkovo1995/Gauss-Kruger13N >
Scale 1: 2,459
Working
List 1:2000
eni_zdanie

Feature Info: Leninsky Rayon

eni (1 feature - 1 selected)

.	gid	index	eni_n	ВИД_НЕД	Адрес_ЕНИ
	9,019	9,260	NULL	NULL	Туюкская 187

Figure 13: Kosmo interface viewing cadastral data stored in PostGIS.

GRASS

The Geographic Resources Analysis Support System commonly known as GRASS is an extremely powerful GI analysis system. While it uses its own internal data format for storage, it can read PostGIS data. It currently runs well under the Linux operating system and while a native windows version has just been released it is not yet considered stable, although it is looking very promising.[5]

However, in the future GRASS will be a major player, as, because of its modular nature, a programme like QGIS is able to make use of its functionality. When this is combined with the functionality available in PostGIS, it is evident that a potentially powerful GI system could be build using these three programmes.

5 Since writing this a stable windows verson of GRASS 6.3 has been released.

4.2.4 Summary of Implementation

The test area was KSUCTA's main campus on Maldybaev Street. There were two test databases involved, one small scale and one medium scale. The tools used were UMN's MapServer, PostgreSQL/PostGIS, Quantum GIS, Kosmo, gvSIG and ArcGIS. This summary is depicted graphically in Figure 14 below

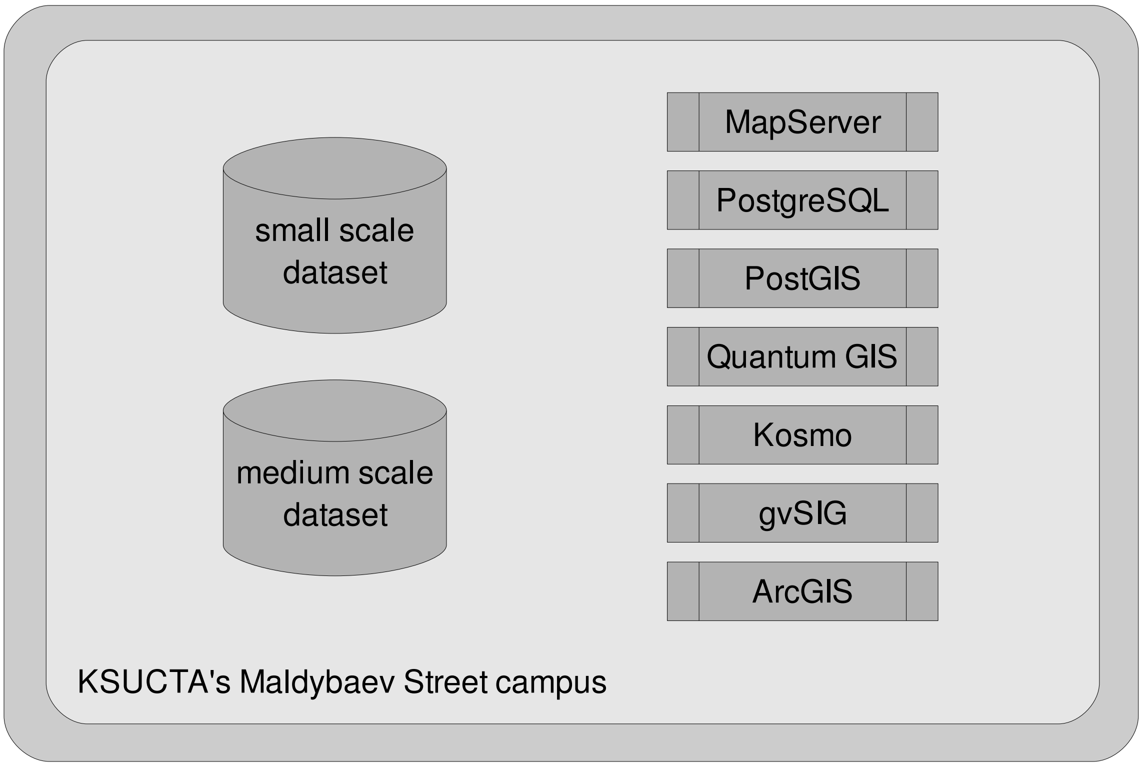


Figure 14: Diagram showing the implementation components of a low cost client/server GIS.

The previous section (4.2.3 Tools) has shown that it *is* possible to serve geographic data to a web browser, as well as other GIS clients, using only open source software and that doing so does not require an in–depth IT or web programming knowledge. This is important for the GI professional, whose concern is the data rather than the technology. However, the work detailed so far was not the final solution. The MapServer interface itself still needed considerable adjustment and issues of data indexing in the database, query response times, managing map projections and coordinating systems etc. still needed to be investigated. These and other issues are discussed more fully in the next two chapters.

CHAPTER 5 RESULTS

The previous chapter was quite detailed in focusing on the selection of data and tools to be used in the building of the prototype. This chapter takes a step back, looking at what was expected at the beginning of the project, see section 1.4 Expected results, and then comparing the expectation with what was actually achieved. In doing this it simply presents the results with very little discussion, as a detailed discussion takes place in Chapter 6 Analysis of Results which follows.

5.1 The specification

The specification for a low cost client/server geographic information. This will be at quite a conceptual or high level but will deal with standards, and with platform and technology issues.

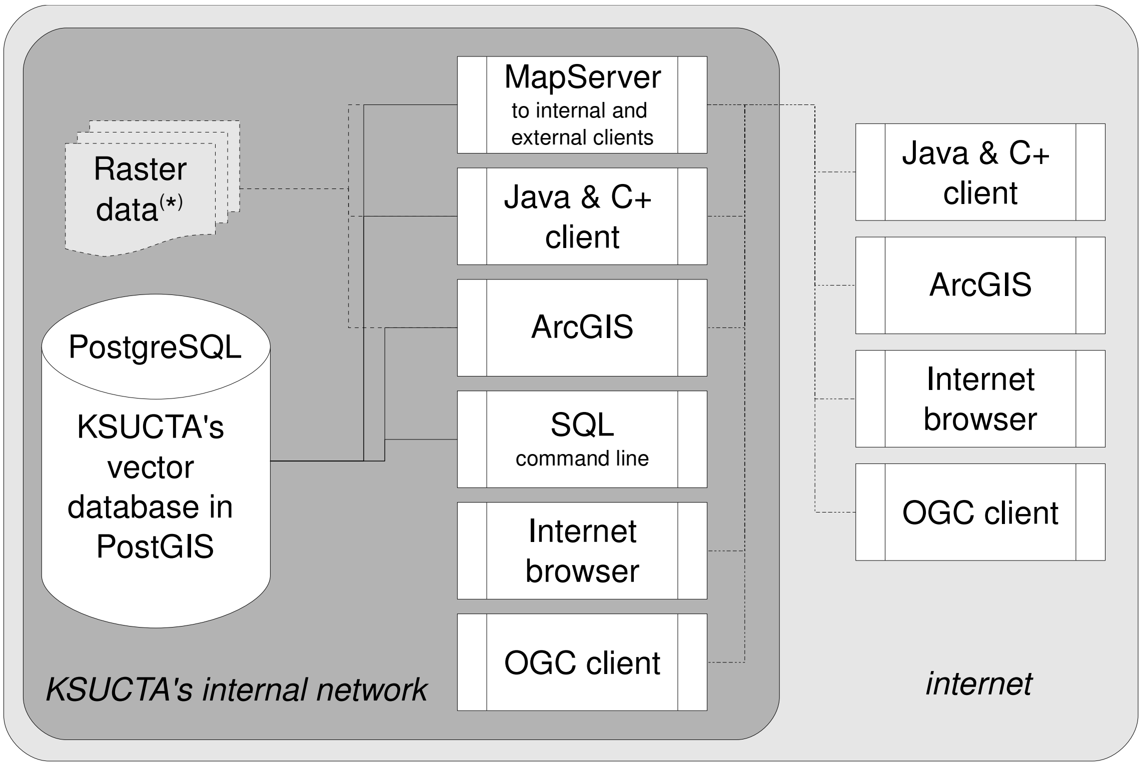


Figure 15: Diagram showing the specifications for a low cost map server.

(*) Note that, as raster data currently cannot be served from PostGIS, it is necessary to store this on the file system and make the folder available to the client applications.

However, MapServer can still serve raster data very efficiently to a web page client and to its ArcIMS emulator.

Figure 1 in section 1.3.1 shows a conceptual diagram of how the system might operate but, for the specification, a level of detail was needed that dealt with the specific technology to be used. Therefore, following preliminary work, detailed in section 4.2.3, PostGIS was chosen as the database server and MapServer as the web server. PostGIS would also allow access to the database from a Java or C+ client like QGIS, an ArcGIS client via the “Data Interoperability” extension, and directly from command line SQL. Along with a pre-designed web interface, MapServer would make the data available as an ArcIMS service and in a way that adheres to OGC specifications - see Figure 15 above.

Figure 15 also shows a clear delineation between KSUCTA's internal network and the internet and provides only one path, i.e. MapServer, between the two. This is important as data security over the internet needs to be managed on this link alone, while the single link to PostGIS manages internal and overall data security.

5.2 The requirements

Knowledge of the requirements of an institution that can be met by such a system. It is these requirements that will detail such issues as the amount of data involved, the loading on the system (i.e. the number of concurrent users) and bandwidth requirements. This will also include any limitations that there may be in comparison to a commercial solution.

Diagrams are easy to draw and conceptualise, but the implementation needs to take account of on-the-ground issues and limitations. Specifically, two issues will be dealt with here: data quantity and concurrent access capabilities.

5.2.1 Quantity of data

Ideally, there would be in existence complete coverage of vector topographic and cadastral data, to form a base on which other data could be overlaid. Even though, such data coverage is not currently available, it still needs to be allowed for, as these two themes could well comprise 80% of the total database in the future.

The estimation of data quantity was achieved by considering the data that is available, mainly around Bishkek and the Chui Valley, and extrapolating that coverage to the whole country, in order to obtain a figure representing megabytes of storage. For this purpose only data layers which require large amounts of disk storage space have been considered.

The current extent of topographic data comprises four 1:50000 map sheets around the city of Bishkek, as shown in Figure 4 on page 23. The size of this topographic study area is 1520km^2 compared with 199470km^2 for the territory of Kyrgyzstan, i.e. the topographic study area is 1/132 the size of the whole country. This ratio was used to extrapolate data considered to be distributed uniformly across the country, e.g. rivers, relief etc.

Data such as roads and urban blocks needed to be extrapolated in proportion to population. This process was a little more complicated, and was done by counting the number of features, both inside and outside the urban zone, within the topographic study area and then extrapolating both these figures across the whole country. The ratios, or multipliers, used for this extrapolation were obtained by considering the urbanised area within the topographic study area, approximately 167km^2, and comparing it with the urbanised area across the country as a whole, approximately 930km^2. Table 3 shows a summary of these figures.

Table 3: The multipliers used for interpolation of data storage requirements

	within the topo-study area	across the whole country	multiplier used for interpolation
urbanised area	167 km^2	930 km^2	5.6
non urbanised area	1353 km^2	198540 km^2	147
total area	1520 km^2	199470 km^2	132

Table 4 below shows the space requirements for topographic and cadastral vector data along with raster data currently held by KSUCTA.

Table 4: Data storage space requirements

Data layer	number of features in topo-study area	MB in topo-study area	MB interpolated across the country	Comments
Vector topographic data				
contours	4528	3.0	396.0	{u}
roads	12387	2.1		{p}
- urban	6722	1.14	6.4	
- non urban	5655	0.96	141.0	
urban blocks	5151	1.7		{p}
- urban	3378	1.1	6.2	
- non urban	1773	0.6	88.0	
rivers	3850	1.2	159	{u}
populated pl.	4610	1.0		{p}
- urban	1437	0.3	1.7	
- non urban	3173	0.7	103.0	
lakes	424	0.4	53.0	{u}
Total vector topographic			*795.3*	
Vector cadastral data				
land parcels			2048.0	based on an estimate of 1 million land parcels (approx 2GB)
Total vector			***2843.3MB***	
Raster topographic data				
scanned map	4	160	22.0GB	{u}
Other raster data				
Landsat TM				
colour composite images			2.5GB	12 existing images
individual bands			15.0GB	20 existing images
SRTM (DTM)			0.5GB	7 existing images
Total raster			***40GB***	size of uncompressed TIFF files

{u} = extrapolated uniformly {p} = extrapolated in proportion to population

The above figures do not represent an exhaustive list and there will be additional data added to the database from time to time. However, for the purpose of this study, they served as a good minimum benchmark for the system to operate on.

5.2.2 Concurrent access to the data

Access to the data through the map server is delivered to PC client computers, either using a specific GIS client, e.g. ArcMap/ArcExplorer, an internet browser, or other OGC compliant client.

KSUCTA has currently about 700 PC workstations in 40 separate classrooms/laboratories on its local area network. Not all of these will access the map server at any one time and the actual number may in fact be very small. For the purpose of this study, a figure of 5%, or 35 PCs, was chosen as the loading that the map server should be able to manage simultaneously.

5.3 A prototype

Building a prototype. This will be a working model that is able to demonstrate the full capabilities of the system to fulfil the requirements.

Building the prototype involved loading the data into a number of databases, installing the chosen client applications and developing an enhanced web page application based on UMN's MapServer. A test situation was provided during a training seminar, for which an additional 16 computers had been temporarily installed, with both ArcGIS and a variety of open source clients. With the existing 25 in the Department of Geoinformatics there was enough to perform a realistic trial. The results of this trial are introduced in section 5.3.2 and detailed in chapter 6, Analysis of Results.

The following sections overview the databases created, and introduce some of the issues encountered. A more detailed coverage of these and other issues encountered while building and assembling the system can be found in chapter 6.

5.3.1 Databases created

Three databases were created on PostgreSQL one to contain an increased overview level dataset, one to contain the topographic data detailed in Table 1 on page 23, and one to hold a selection of cadastral data for part of Bishkek, which was provided by the State Land Registry Office for the purpose of the training seminar mentioned above, and was only available for this period. These databases are listed in Table 5 below.

Table 5: Databases created as part of the prototype

Dataset name	Scale of data	Projection of data	Coverage of dataset
Kyrgyzstan_1mil	1:1 000 000	WGS 84	Kyrgyzstan
Bishkek_topo	1:50 000	GK 13	Bishkek and peri-urban area
Bishkek_cad	< 1:5000	GK 13	Leninsky District of Bishkek

5.3.2 The testing process

During the training seminar mentioned above, necessity drove an impromptu trial of our system, when participants from the State Land Registry Office, supplied their own data for sessions which were to cover open source GIS. These data, which were supplied in shape file format, used Cyrillic type for some attributes and also for field name definitions. Viewing this data either in ArcView, Quantum GIS, or Kosmo proved problematic in the correct display of attribute information. This was overcome by simply loading the data into PostgreSQL on the server. This meant that, for the first part of the course, there were 17 users[6] simultaneously accessing the data in PostgreSQL.

This impromptu test was not completely without problems as the data, which comprised mainly parcel, road, and services data, representing a quarter of Bishkek, (comprising approximately 30 000 parcels) had simply been loaded to the database without the creation of any spatial indexes. As a result response times, when zoomed to the whole extent, were between 5 to 10 seconds. However, when zoomed in to a more

6 16 students + the instructor

appropriate scale for cadastral data, i.e. less than 1:5000, response times were always sub-second regardless of how many people were accessing the data at the same time. Additionally, the posting back of edits to the database was also very fast, as only parcel/s being edited were sent, as opposed to editing a shape file where the whole file is rewritten for each save and other users are locked out for the duration of the editing session.

At the completion of the training seminar a decision was made to do a full test and to try to *crash* the system by loading it up with an additional 25 users. This meant that over 40 users were simultaneously accessing the database, often using more than one application per computer, i.e. using say both Quantum GIS and an internet browser at the same time. Response times were on the whole good, especially when only using vector data within PostgreSQL. The exception to this was direct access of the raster data stored on the file system, as opposed to accessing it through MapServer as an OGC service. These results are discussed in more detail in the next 2 chapters.

CHAPTER 6 ANALYSIS OF RESULTS

Each of the result areas, listed in section 1.4, proved to be a moving target. Firstly, because of changes in technology, even during the duration of this study, and secondly, because they are all interrelated and a change in one effected a change in another. An example of this is the Java client Kosmo. Initially, this had a serious bug with handling queries involving Cyrillic type, which rendered it not suitable for use in our situation. However, after contact with the developers via the "support" discussion list, the problem was addressed and put right with the next release. Other on-going changes came about because GI science is still quite new in Kyrgyzstan and is still being "found out about". Thus, as new people are seeing its advantages, they are wanting to "come on-board" and be a part of it, hence the specification changed after the project was already running.

Because of the "moving target" nature of these result areas, the results presented in the previous chapter reflect what was finally arrived at and decided upon at the beginning of May 2008. There have been, however, further developments and these are detailed in the next chapter in section 7.2 Future work. The remainder of this chapter details issues that arose in relation to the data and to software implementation, and to the implementation of the OGC services.

6.1 Issues arising

Following the initial work carried out, as described in section 4.2 Implementation, the following issues were identified as areas needing resolution for any prototype to be viable:

Issues involving PostGIS

- Username/passwords and roles
- Data location
- Backup and restore

Issues involving MapServer

- Passwords in ".map" file
- Coordinate transformations in MapServer
- Webpage interface design

- Automation of ".map" file creation

Issues involving Data

- Extents of the database
- Preparation of raster data
- Data indexing

The resolution of many of these issues required a much deeper look into the tools already discussed in section 4.2.3 Tools. This meant taking the analysis beyond simply stating what should be possible in theory, to working specifically through the issues of what was actually required to make it happen. Also addressed here are some of the issues involved with moving from a development to a production environment, where issues of security and integrity are important if the resultant system is to be viable and robust.

6.1.1 Issues involving PostGIS

Username/passwords and roles

“Out of the box” PostGIS creates and populates databases using the DBA (or database administrator) privileges. Whilst it is possible for multiple users to view and edit the data in a client like QGIS using these privileges, it was not desirable for general use and, therefore, it was necessary to implement a system of access roles.

Roles are easily managed using the pgAdmin tool shown in Figure 9 of the section on PostGIS and PostgreSQL on page 30. The procedure to provide for read only, or select only, access for general users involved two steps. Firstly, to create a new login role and then to grant “select” access to the required tables using the “Grant Wizard”. It should be noted that for these tables to be available in a GIS client like QGIS the OGC meta-data tables “geometry_columns” and “spatial_ref_sys” need also to be made available to the new role, again with “select” access.

Data location

Unlike Oracle, when PostgreSQL is installed the user is not prompted for the physical disk location of the database cluster[7] . On the windows platform this database cluster is, by default, created on the system or “C:” disk. This was not a desirable situation, as it was intended to have the database residing on a drive specifically partitioned for this purpose.

A database cluster is a single directory, or folder, under which PostgreSQL stores all databases. This directory is pointed to by the system variable, PG_DATA[8]. PostgreSQL provides a relatively simple process to move this folder to a different location:

1. use the utility “initdb” to create a new database cluster
2. edit the configuration file “postgresql.conf” to point to this new location
3. stop and start the “PostgreSQL Database Server (Postmaster)” service
4. finally recreate the database and load the data from the command line detailed in Listing 16 of Appendix C

Usually this process would be part of the setting up procedure before any database is created and is a one-off process.

Backup and restore

The ability to make regular backups is vital to any IT database project. PostgreSQL offers a backup and restore facility, but this facility needed to be fully tested to ensure that it was suitable for KSUCTA’s situation. In particular, it was necessary to know about issues of

- Type of backup available e.g. incremental, differential or full backup

7 A database cluster is the physical location on the hard disk where subsequent databases are created.

8 The default location of this directory after installation is “Program files\PostgreSQL\8.2\data”

- Whether a “warm backup” is possible, i.e. is a backup possible when the system is still serving and receiving data, or is it necessary to shut the system down for backup.
- What is involved in the restore process, i.e. can a backup be restored simply? In addition, if it needs to be restored to a different server, what additional operations are involved and how much time is required to have the system functioning normally.

Postgres offers three different approaches to backup: SQL dump, File system level backup and Continuous archiving. Each has its own strengths and weaknesses.

SQL Dump

SQL dump is run from the command line and writes the result to standard output (which can be piped to a file)

```
pg_dump dbname > dmpfile
```

This creates an internally consistent output, i.e. updates to the database while pg_dump is running will not be in the dump. Also, pg_dump does not block other operations on the database while it is working, with the exception of those operations that themselves, need to operate with an exclusive lock.

Restoring a dump is also a simple operation, again from the command line

```
psql dbname < dmpfile
```

The new database should have been created with only the default template, that is not using the POSTGIS template, as all languages procedures etc. will be in the dump file.

Finally, it should be mentioned that, because of the ability of pg_dump and psql to write to or read from pipes, it is possible to dump a database directly from one server to another; for example:

```
pg_dump -h host1 dbname | psql -h host2 dbname
```

"File system level" backup

This method basically involves backing up or copying the files and directories that make up the database cluster. While this method is simple and easy to implement, it does require an exclusive access to the database to ensure that the backup is internally consistent.

Continuous archiving

PostgreSQL maintains a write-ahead log (WAL) meaning that it is possible to perform a restore to any point in time by restoring a "file system level" backup and then replaying the log entries to any point we choose. This, however, is somewhat complicated to set up and can result in a restore that in not internally consistent.

The SQL dump is definitely the preferred method, as it offers an internally consistent result that is easy to implement and it can be scheduled on a regular basis or performed interactively using the pgAdmin tool.

6.1.2 Issues involving MapServer

Passwords in ".map" file

As shown in Listing 17 of Appendix C the amendments made to the map file to enable MapServer to access PostGIS databases required that the password be included in a non-encrypted form. This does not seem to be an appropriate practice as the ".map" file could be publicly visible. It is possible that the ".map" file could be stored in a area not accessible to the internet, although a better and far more preferable solution would be, that the password and, possibly, the username to be obtained from the user.

Investigations carried out on how MapServer can be configured to respond to OGC requests provided a way for the ".map" file to be hidden from the public. This is documented in Listing 18 of Appendix D. Such an approach, however, is not really called for here as a specific role, which grants only "select" or read only access to the database, can be created so that knowledge of this username and password need not be an issue for data security.

Coordinate transformations in MapServer

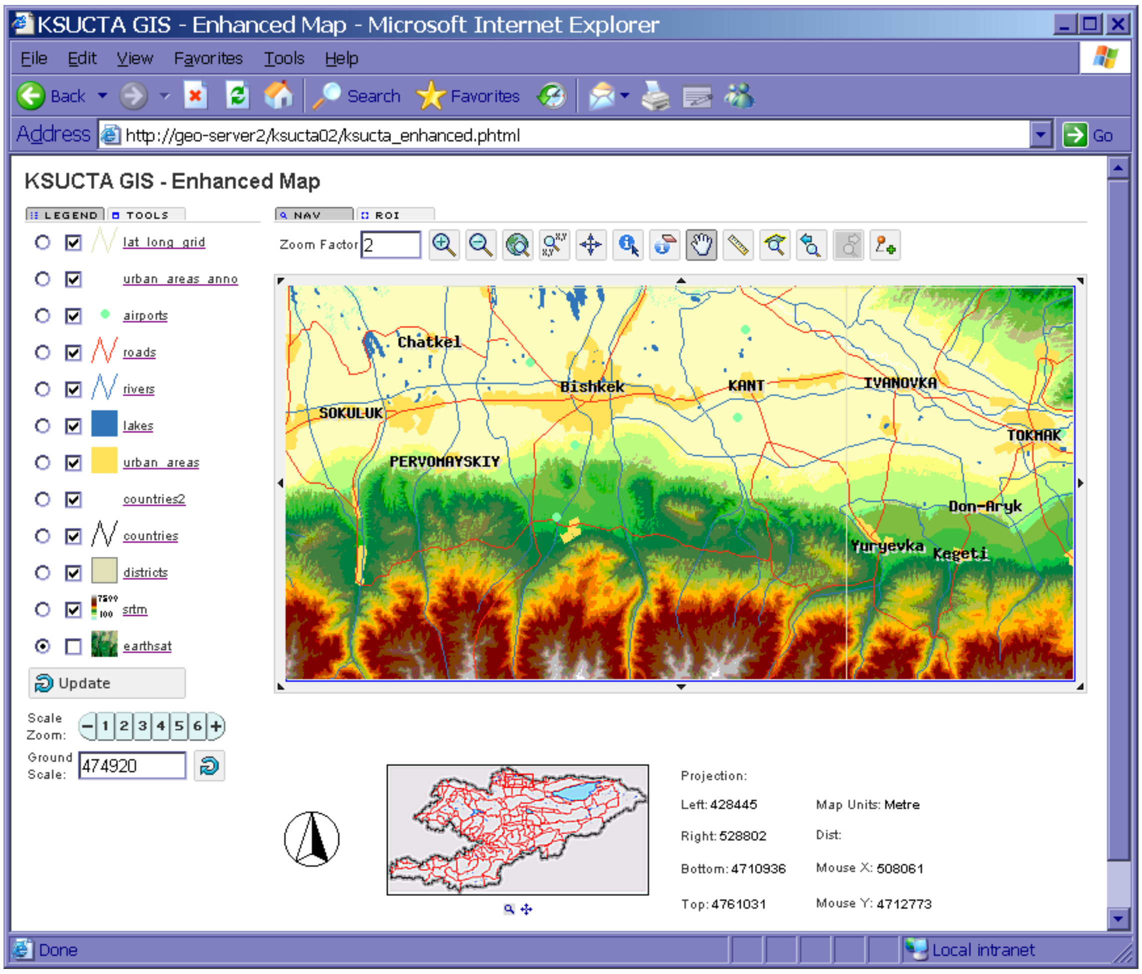
As mentioned in the section on Map projections and coordinate systems on page 26, it is desirable to store all data in its native coordinate system and then re-project on-the-fly as necessary. This is possible in MapServer, by altering the relevant sections of the ".map" file. ArcGIS and the three open source clients mentioned each have the ability to handle and convert between the required map projections. Appendix F lists the map projection capabilities of the programmes tested, as well as other open source GIS programmes.

Webpage interface design

The MapServer interface shown in Figure 8 in section 4.2.3 was just to show the idea of how straightforward web mapping is in theory, but it was never intended to be a final production solution. The development of a production web interface can be as large, or as small as the developer has time for. For this study, however, it was desirable not to invest a large amount of time into the web interface by itself, although it was necessary that the web interface be robust and of a reasonable quality. For this reason, an add on to MapServer called Chameleon was selected as the platform on which to build the web interface, as it greatly reduced the work needed. "*Chameleon incorporates the ability to set up new applications quickly from a common pool of "widgets" that can be placed in an HTML template file.*" [Redfern 2005]

By using the same ".map" file created earlier (as documented in Appendix B and modified for environment variables, as shown in Listing 18 of Appendix D), it was possible to modify a ready-built sample Chameleon application quickly and produce a far more appealing and stable web interface, this also included some basic analysis tools like measuring, searching, querying, map layout capabilities, a map resize facility, provision for users to choose their own map projection to be done on the fly, as well as the ability for users to add their own points as map notes.

Chameleon offers a huge number of possibilities, and will definitely be used in future web mapping development at KSUCTA, but for the purpose of this study the standard interface described in the previous paragraph and illustrated in Figure 16 below, was considered sufficient.



http://geo-server2/ksucta02/ksucta_enhanced.phtml

KSUCTA GIS - Enhanced Map

Zoom Factor: 2

Scale Zoom: − 1 2 3 4 5 6 +

Ground Scale: 474920

Projection:

Left: 428445 Map Units: Metre

Right: 528802 Dist:

Bottom: 4710936 Mouse X: 508061

Top: 4761031 Mouse Y: 4712773

Figure 16: A screen dump of the basic MapServer app. using Chameleon widgets.

Automation of ".map" file creation

QGIS was billed as being able to create MapServer “.map” files but to do this it needed the installation of Python 2.5. In addition there is a freely downloadable extension for ArcGIS that also creates “.map” files. These were very useful utilities as the creation of a “.map” file can be quite time consuming when it involves a large number of layers, especially when trying to visualise colours and symbols from their RGB numbers.

These tools while good, and easy to use, still require some manual intervention in that the resultant ".map" files need a degree of tweaking – e.g. to correct path names, single to double quotes etc. These changes were a little inconsistent in where they were needed but generally were quite minor compared with having to work out out the symbology section of the file using RGB numbers.

6.1.3 Issues involving Data

Extents of the database

A simple but necessary step was to decide on the overall extents of the database. To cater for cross border issues and in order to create visually appealing maps it is necessary to have data extending beyond the borders of the country. Figure 17 shows the chosen extents between latitudes 38 and 44 degrees north and longitudes 67 and 81 degrees east.

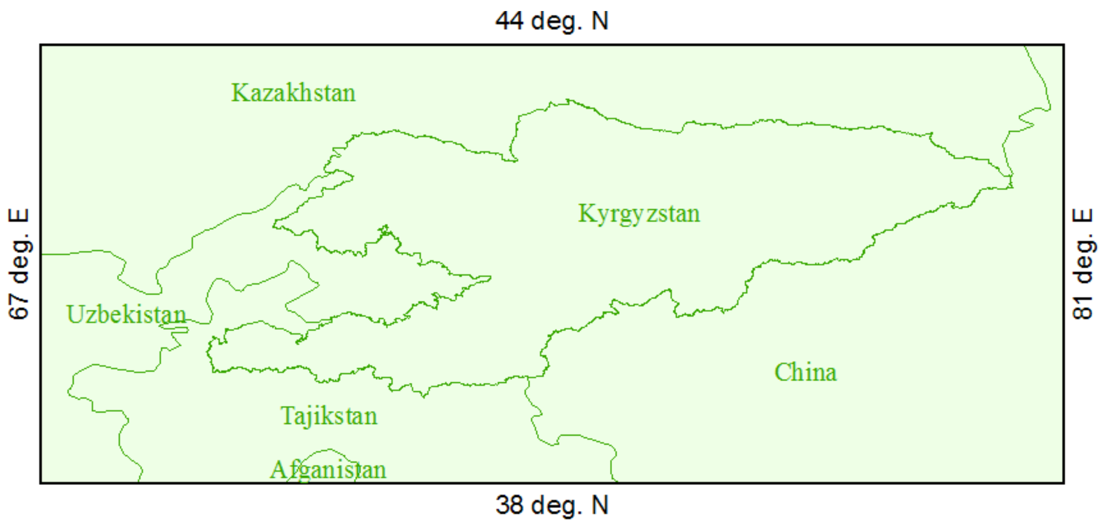


Figure 17: Map showing the chosen extents of KSUCTA's database.

Preparation of raster data

As PostgreSQL currently provides no facility for the storage of raster data, this data was stored in a folder on the file system and served via MapServer. Although these data could be accessed directly by most GIS clients it was desirable that they be delivered from MapServer as a WMS or an ArcIMS service, thus reducing network traffic. From loading trials it was determined that MapServer performed best serving uncompressed TIFF images of between 100Mb to 200Mb in size. Table 6 below lists the tile pattern used, The EarthSat image was left untouched, while the SRTM data was simply split in half vertically. The Landsat colour composites were in UTM projection and spanned three zones. Therefore, the tile pattern also spanned three zones, as shown in Figure 18 below.

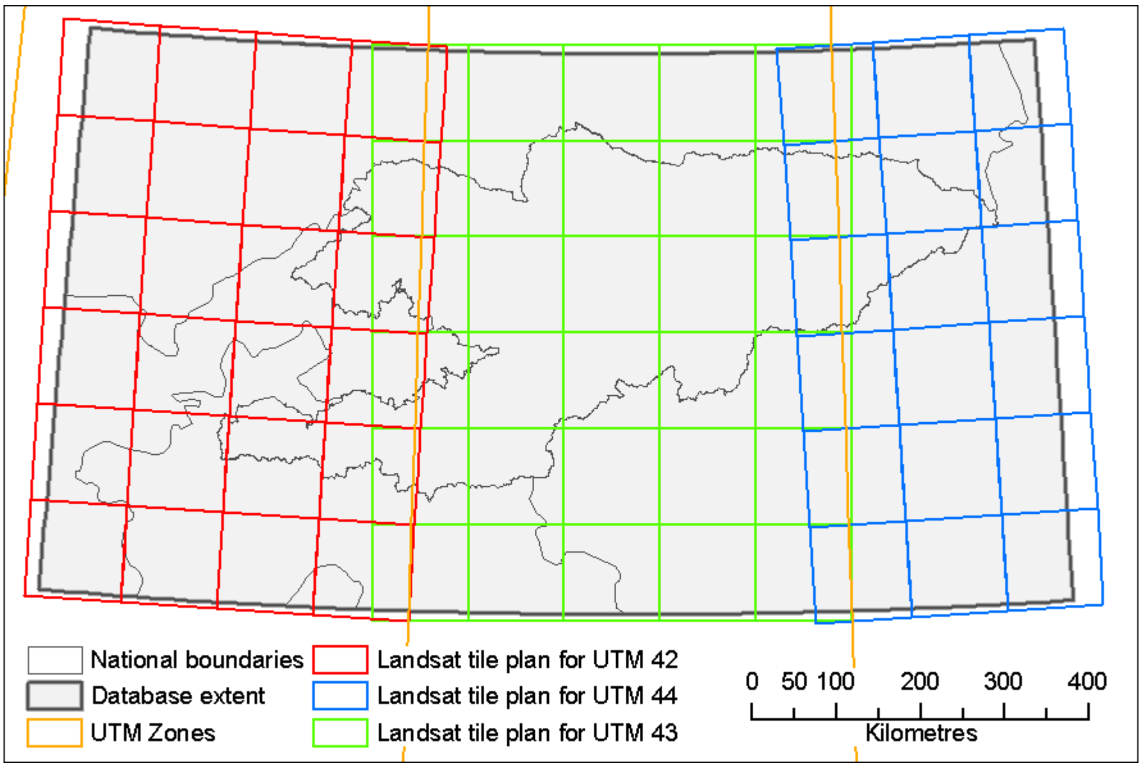

National boundaries
Database extent
UTM Zones
Landsat tile plan for UTM 42
Landsat tile plan for UTM 44
Landsat tile plan for UTM 43
0 50 100 200 300 400
Kilometres

Figure 18: Map showing tile pattern for Landsat TM images.

Table 6: The tile system for raster data

Raster dataset	Pixel size	Projection system.	Tile size (km)	Tile size (Mb)	Number of tiles
EarthSat	150m	WGS 84	1200 x 670km	130Mb	1
SRTM	90m	WGS 84	600 x 670km	180Mb	2
Landsat colour composite	15m	UTM zones 42, 43, & 44	114 x 114km	190Mb	72

Data indexing

As mentioned in section 5.3.2 above, when the drawing the cadastral data, while the view is set to the full extent, the response time was unacceptably long. While indexing the data, both on spatial and attribute columns and using other data optimisation utilities provided by PostgreSQL and PostGIS will speed up certain query and analysis functions, the issues discovered during the testing relate to network traffic issues and are more a management issue rather than a data structure issue. This is discussed further in section 7.2.

6.2 The implementation of OGC specifications

The desire to adhere to OGC standards was stated in section 1.5 Constraints and risks, and in section 3.1.2, the three standard OGC services of WMS, WFS, and WCS were specifically mentioned as being desirable for implementation in this study.

These are all supported by the MS4W build of MapServer and implemented simply by adding OWS (OpenGIS Web Services) directives to the ".map" file.

OWS offers a unified approach to metadata using "ows_*" type directives, this means that instead of many declarations like:

```
“wms_title” “Land Use”
“wfs_title” “Land Use”
```

A simple:

```
“ows_title” “Land Use”
```

will satisfy all *title directives (e.g. [wms|wfs|wcs|gml|sos]_title) However, it is still possible to set service-specific metadata if needed, which overrides "ows_*". See Appendix E for a fuller discussion of how the ".map" file is amended to include the OGC directives.

OGC services are also supported by the three open source clients chosen for the study and by ArcGIS by using the "Data Interoperability" extension. Appendix F lists the exact OGC capabilities of the programmes tested as well as other open source GIS programmes.

CHAPTER 7 SUMMARY, DISCUSSION AND FUTURE WORK

7.1 Summary and Discussion

This thesis, which started as a feasibility study for building a low cost client/server GIS to meet the requirements of the Kyrgyz State University of Construction, Transport and Architecture (KSUCTA) has implications far beyond those of a single university. Many of the issues identified herein, such as cost factors and data infrastructure, apply also to other institutions and government agencies across the developing world.

There has been in Kyrgyzstan, as in many emerging countries, a growing awareness of GIS by educationalists and scientific professionals and a desire to use GI science to carry out their work more efficiently. Unfortunately, this desire has often manifested itself in the ad hoc gathering of data sets which are kept inaccessible to the wider community, partly because of the insufficient infrastructure to make them accessible, and partly because information is viewed as control, which no one wants to relinquish. Additionally, a lot of what has been done has utilised unlicensed, or pirated, software which, taking a long term view, is not sustainable for a country wanting to develop. This study has sought to investigate technologies that can contribute to, and even form the backbone of, a national spatial data infrastructure (NSDI).

Two important tenants in the approach to this study were the use of Open Source software and the desire to adhere to Open GIS, or OGC, standards, and these need to be commented on briefly. Firstly, the cost advantages of open source should not diminish the fact that it has actually reached a level of maturity where it rivals many commercial solutions, so that it is being selected not just on price but on its functionality. Secondly, the adhering to OGC standards does not mean merely paying lip service to something generally viewed as a good idea, and it also does not have to be a burdensome task. The tools selected do all the work of reading and writing to OGC specifications, mean that all the user needs to do is to configure the relevant interfaces and configuration files and provide the required level of meta-data. If what has been developed here is to be used sustainably outside the KSUCTA environment, these two principles will be fundamental.

Of the tools discussed in this study the most important is PostgreSQL and PostGIS, which spatially enables it as it forms the database where all the data is stored. It is, therefore, important to know exactly the level of compatibility each of the clients have to it. That is, whether they can read and write directly to it, like Quantum GIS, or do they just convert the data to their own format or cache the data somewhere in some intermediate format, as ArcGIS seemed to do when using the data interoperability extension. Likewise, for future expansions to the system, it is important to know the level of compatibility to OGC standards. The difference between a full and direct compatibility and a partial data reading/import/export ability will greatly affect the speed of real time operation of the system.

The quantity of data was not a problem, as modern day data storage costs are small. However, what was a problem was network traffic volumes for large vector datasets. Where a request was made for the full extent of a large vector layer to a direct access internal client such as Quantum GIS or as an XML file to a WFS, there was a significant delay in response due to network limitations. Issues of network traffic also played a part when accessing large raster datasets via an internal client and not via a MapServer WMS.

The concurrent access to the map server proved not to be a problem when testing the 35 simultaneous connections mentioned in section 5.2.2 and when viewing data at an appropriate scale. The system could probably have handled twice the loading without showing any major problems.

Although issues did crop up and are documented in chapters 5 and 6 and although there is still work to do, see section 7.2, it has been shown that it is perfectly feasible to build a client/server GIS that utilises open source technologies and hence does not require a large software budget. However, whether this solution could be called a “Low Cost Client/Server GIS” is not clear, as such systems do require significant human resource in both the development and the organisation aspects of the project. If, because it is unpaid, this human resource is seen as low or no cost, it does tend to under-value the project as a whole. It might, therefore, be better to sum up by saying "It is possible to use low cost technologies to support the building of a client/server GIS in the developing world".

7.2 Future work

A number of issues were revealed during the course of this study, some of which were able to be resolved and others not. Some issues however, were considered significant enough to be flagged for future attention. What follows is a brief description of the more important of these issues, some of which are purely technical and therefore, relatively straight forward while others are organisational and political and will not be so simple.

7.2.1 Transaction size limit

The issue highlighted during the trial of a very slow response time while drawing up the full extents of a large vector layer was more an issue of work flow management rather than data structures. It should not be necessary to draw up every cadastral parcel in Bishkek City at a single time, but an inexperienced user might easily add this layer while zoomed out to the full extents of the city. Likewise, a WFS may request an XML dump of the whole layer. While both these requests to the database are valid, they could place an unacceptably high load on the network to deliver these services to the user. Therefore, the possibility should be investigated of limiting the size of transaction permitted.

7.2.2 OGC and GeoServer

MapServer, while an excellent choice for creating a web mapping interface, struggled somewhat at serving the full suite of OGC services, e.g. it is not able to preform the editing functions of the WFS, see Appendix E. There is another product called GeoServer, available from http://geoserver.sourceforge.net/, which excels at serving OGC services. It is the intention of KSUCTA to explore the feasibility of using GeoServer to serve OGC services in the final quarter of 2008.

7.2.3 Oracle Spatial

KSUCTA has, as part of a software development and training project that is not connected with this study in any way, obtained a site licence for Oracle. While it is obviously not open source, and therefore, not strictly in keeping with the main goals of this study, there are a great many reasons for considering Oracle Spatial as the back end database engine. One reason in particular is its ability to store and serve raster data. It is, therefore, intended to explore this option further as the database engine and to compare capabilities and performance with PostgreSQL.

7.2.4 Promotion of OGC principles

As indicated in section 1.6, it was not the intention of this study to obtain political commitment to Open GIS or to the building of a national SDI. However, these issues need to be promoted to the most senior level of government in a way that is able to overcome fears about loss of control when data and information are put in the public domain. It needs to be pointed out that during the course of this study, contact was made with people at an operational and lower management level in various state organisations. Almost without exception these people were able to see quickly the advantages of information-sharing for scientific purposes, and it will be important to encourage those in senior management and government ministerial positions to recognise and implement those advantages.

7.3 Concluding Statement

In conclusion, it has been shown that it is technically possible to build a “Low Cost Client/Server GIS” but, in reality, real and genuine support from the upper levels of central government and from the university hierarchy would be needed if such a system were to be properly implemented in Kyrgyzstan.

References

CGIAR Consortium for Spatial Information, 2007, "Home page of CGIAR-CSI" , http://srtm.csi.cgiar.org/SRTMdataProcessingMethodology.asp (accessed 1 June 2008)

DM Solutions, 2007, "MapServer for Windows" , http://maptools.org/ms4w/ (accessed 8 November 2007)

Earth Satellite Corporation (EarthSat), ESRI. "Metadata description for the 'Asia West EarthSat NaturalVue Global Landsat Mosaic'." ArcGlobe metadata 2004:

ESRI/US Defense Mapping Agency (DMA), 1993, "About the Digital Chart of the World Data Server" , http://www.maproom.psu.edu/dcw/dcw_about.shtml (accessed 13 September 2007)

Fawcett D & Butler H, 2007, "MapServer New Users - Information to help a New User get going with MapServer..", , Chpt. 3

Garretson C & Fontana J, 2005, "Real deal: Deploying open source in the enterprise.", Network World, July 2005 edition,

INSPIRE: European Union, DIRECTIVE 2007/2/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 14 March 2007establishing an Infrastructure for Spatial Information in the European Community (INSPIRE), 2007

Jantaev M & Zubovich A. "Scope of using OGC Compliant Open Source Sortware for Developing Central Asian Geodatabases."Proceedings from the "Second Central Asian GIS Confrence - GISCA 08" entitled "GIS for the future of Central Asia" 2008:

Karypov A & Kazkanova K. "Open Source Solution for Land Cadastre Systems in Kyrgyzstan."Proceedings from the "Second Central Asian GIS Confrence - GISCA 08" entitled "GIS for the future of Central Asia" 2008:

Kralidis T, Assefa Y, Doyon J F, Lime S, Morissette D, McKenna J. "MapServer OGC Web Services Workshop." 2007:

Mitchell T, 2004, "Open source GIS proves its maturity with excellent conference" , http://www.oreillynet.com/digitalmedia/blog/2004/06/open_source_gis_proves_its_mat.html (accessed 30 October 2007)

NASA, 2005, "Geospatial Interoperability (GI) Return on Investment Study Report.", , pg. 4

Nazarkulova A B. "A conceptual model for cross-border NP SDI node."Proceedings of the International Confrence on Geodesy and Geoinformatics 2007 :206-212

Open GIS Consortium, Inc., 1999, "OpenGIS Simple Features Specification for SQL - Revision 1.1.", , pg. 1

Open Source Initiative, 2007, "The Homepage of the Open Source Initiative" , http://www.opensource.org/ (accessed 3 November 2007)

Perry M & Mosheh G, 2005, "HostGIS Linux - A MapServer-oriented Linux distribution.", , pg. 2

Redfern D. "Chameleon Application Developer's Guide - version 2.0." Chameleon Documentation 2005: Introduction

Refractions Research, 2005, "PostGIS homepage" , http://postgis.refractions.net/ (accessed 29 January 2008)

Sutton T, Blazek R, Holl S, Dassau O, Mitchell T, Morely B, Luthman L. "Quantum GIS User Guide, Version 0.8 ’TITAN’." 2006: Forward (pg. 1)

Simpson, John E, 2004, "Mapping and Markup, Parts 1 & 2" ,
http://www.xml.com/pub/a/2004/11/24/tourist.html (accessed 6 February 2007)

Smits P C, 2002, "INSPIRE Architecture and Standards Position Paper.", , pg. vi

Steiniger S, 2008, "OGC Functionality" , http://www.spatialserver.net/osgis/ (accessed 20 May 2008)

University of Minnesota, 2007, "Welcome to MapServer - UMN MapServer" ,
http://mapserver.gis.umn.edu/ (accessed 8 November 2007)

Wikipedia, 2008, "Open Source" , http://en.wikipedia.org/wiki/Open_source (accessed 4 March 2008)

Zeiss G. "Open Source and Interoperability."Map Africa Confrence 2006 2006:

Appendix A - Description of OGC Services

Following are brief descriptions of the three OGC services referred to in section 3.1.2 Open GIS and the subsection on “The implementation of OGC specifications” in section 6.2

Web Map Service

Taken from the “Overview” on http://www.opengeospatial.org/standards/wms

The OpenGIS® Web Map Service (WMS) Implementation Specification provides three operations (GetCapabilities, GetMap, and GetFeatureInfo) in support of the creation and display of registered and superimposed map-like views of information that come simultaneously from multiple remote and heterogeneous sources.

When client and server software implements WMS, any client may access maps from any server. Any client may combine maps (overlay them like clear acetate sheets) from one or more servers. Any client may query information from a map provided by any server. While programmers need to write code to implement the specifications, end users can take advantage of products that include them to publish and access geospatial information. Software buyers can choose the best solution for their needs and not worry about if it will work with other solutions; if they all implement the same standard, WMS, they will all work together.

In particular WMS defines:

- How to request and provide a map as a picture or set of features (GetMap)
- How to get and provide information about the content of a map such as the value of a feature at a location (GetFeatureInfo)
- How to get and provide information about what types of maps a server can deliver (GetCapabilities)

Web Feature Service – Overview

Taken from the “Overview” on http://www.opengeospatial.org/standards/wfs

The OpenGIS® Web Feature Service (WFS) Implementation Specification allows a client to retrieve and update geospatial data encoded in Geography Markup Language (GML) from multiple Web Feature Services. The specification defines interfaces for data access and manipulation operations on geographic features, using HTTP as the distributed computing platform. Via these interfaces, a Web user or service can combine, use and manage geodata -- the feature information behind a map image -- from different sources.

The following WFS operations are available to manage and query geographic features and elements:

- Create a new feature instance
- Delete a feature instance
- Update a feature instance
- Lock a feature instance
- Get or query features based on spatial and non-spatial constraints

Web Coverage Service

The following is taken from the Introduction to the document Web Coverage Service (WCS) Implementation Specification

The Web Coverage Service (WCS) supports electronic retrieval of geospatial data as "coverages" – that is, digital geospatial information representing space-varying phenomena.

A WCS provides access to potentially detailed and rich sets of geospatial information, in forms that are useful for client-side rendering, multi-valued coverages, and input into scientific models and other clients. The WCS may be compared to the OGC Web Map Service (WMS) and the Web Feature Service (WFS); like them it allows clients to choose portions of a server's information holdings based on spatial constraints and other criteria.

Unlike the WMS [OGC 04-024], which portrays spatial data to return static maps (rendered as pictures by the server), the Web Coverage Service provides available data together with their detailed descriptions; defines a rich syntax for requests against these data; and returns data with its original semantics (instead of pictures) which may be interpreted, extrapolated, etc. – and not just portrayed.

Unlike WFS [OGC 02-058], which returns discrete geospatial features, the Web Coverage Service returns coverages representing space-varying phenomena that relate a spatio-temporal domain to a (possibly multidimensional) range of properties.

The Web Coverage Service provides three operations: GetCapabilities, DescribeCoverage, and GetCoverage. The GetCapabilities operation returns an XML document describing the service and brief descriptions of the coverages that clients may request. Clients would generally run the GetCapabilities operation and cache its result for use throughout a session, or reuse it for multiple sessions. When the GetCapabilities operation does not return such descriptions, then equivalent information must be available from a separate source, such as an image catalog.

The DescribeCoverage operation lets clients request a full description of one or more coverages served by a particular WCS server. The server responds with an XML document that fully describes the identified coverages.

The GetCoverage operation is normally run after GetCapabilities and DescribeCoverage operation responses have shown what requests are allowed and what data are available. The GetCoverage operation returns a coverage (that is, values or properties of a set of geographic locations), encoded in a well-known coverage format. Its syntax and semantics bear some resemblance to the WMS GetMap and WFS GetFeature requests, but several extensions support the retrieval of coverages rather than static maps or discrete features.

Appendix B - The MapServer “.map” and “.html” Files

The “.map” file:

From the “New Users” guide a “.map” file is defined as *a structured text configuration file for your MapServer application. It defines the area of your map, tells the MapServer program where your data is and where to output images. It also defines your map layers, including their data source, projections, and symbology. It usually has a .map extension.* [Fawcett/Butler 2007] Below is the demo “.map” file with the changes highlighted in bold text that were needed to run with our data, along with some comments.

Listing 1: The start of the ".map" file

The changes in this section don’t require expansion as they are intuitive. The exception is “TEMPLATEPATTERN” which refers to the naming of the “.html” file used to display the data. This is somewhat surprising because this file is specifically named in the “.html” initialisation file. However, it seems that it must also be prefixed with the word “kyrgyz”.

```
#
# Start of map file
#
MAP
  NAME KYRGYZ
  STATUS ON
  SIZE 600 400
  EXTENT 9000 4340000 938200 4790700
  UNITS METERS
  SHAPEPATH "data"
  IMAGECOLOR 255 255 255
  TEMPLATEPATTERN "kyrgyz"
  IMAGETYPE PNG
```

Listing 2: The projection definition

This section had been commented out. However, it becomes important when the data to be displayed is stored in more than one map projection, or if it is necessary to display the data in a different projection to that which they are stored.

```
  #
  # Projection definition, consult the
PROJ.4 documentation for parameter
discussion
  #
  # PROJECTION
  #   "proj=utm"
  #   "ellps=GRS80"
  #   "zone=43"
  #   "north"
  #   "no_defs"
  #
  #   OR:
  #
```

```
#   "init=epsg:32643"
# END
```

Listing 3: The symbols' definitions

This section requires no further explanation at this stage except to state that the standard version of MapServer allows for up to 64 point, line, or fill symbols to be defined.

```
#
# Start of symbol definitions
#
SYMBOL
  NAME 'circle'
  TYPE ELLIPSE
  POINTS 1 1 END
  FILLED TRUE
END

SYMBOL
  NAME 'star'
  TYPE VECTOR
  FILLED TRUE
  POINTS
    0 .375
    .35 .375
    .5 0
    .65 .375
    1 .375
    .75 .625
    .875 1
    .5 .75
    .125 1
    .25 .625
  END
END
```

Listing 4: The WEB section

Again, due to the intuitive use of keywords and variable names, this section requires little comment. It is worth mentioning however, that some of these variables have been defined in the ".html" initialisation file "kg_index.html".

```
#
# Start of web interface definition
(including WMS enabling metadata)
#
WEB
  HEADER templates/header.html
  TEMPLATE "set in kg_index.html"
  FOOTER templates/footer.html
  MINSCALE 250000
  MAXSCALE 9000000
  IMAGEPATH "set in kg_index.html"
  IMAGEURL "set in kg_index.html"
  METADATA
    WMS_TITLE "UMN MapServer Itasca Demo
- altered for Kyrgyzstan"
    WMS_ABSTRACT "This is a UMN MapServer
application for Kyrgyzstan located in
Central Asia."
    WMS_ACCESSCONSTRAINTS "none"
    # change this value to match your
setup
    WMS_ONLINERESOURCE "http://localhost/
demo5/kg_index.html"
    WMS_SRS "EPSG:32643"
  END
END
```

Listing 5: The reference map

This reference map was interesting as it refers to a picture file rather than a layer in the database. Therefore, the “EXTENT” and “SIZE” variable settings are very important if it is to function correctly. Also, the file “reference.png” had to be manually created.

```
#
# Start of reference map
#
REFERENCE
  IMAGE graphics/reference.png
  EXTENT 9000 4340000 938200 4790700
  SIZE 198 98
  STATUS ON
  MINBOXSIZE 5
  MAXBOXSIZE 100
  COLOR 255 0 0
  OUTLINECOLOR 0 0 0
  MARKERSIZE 8
  MARKER 'star'
END
```

Listing 6: The legend

The legend section defines how a legend is to be built. The legend components are built automatically from the data and symbology defined in the layer definition.

```
#
# Start of legend
#
LEGEND
  KEYSIZE 18 12
  LABEL
    TYPE BITMAP
    SIZE MEDIUM
    COLOR 0 0 89
  END
  STATUS ON
END
```

Listing 7: The scale bar

This defines how the scale bar is built and requires no further explanation at this stage.

```
#
# Start of scalebar
#
SCALEBAR
  IMAGECOLOR 0 0 0
  LABEL
    COLOR 255 255 255
    SIZE TINY
  END
  STYLE 1
  SIZE 100 2
  COLOR 255 255 255
  UNITS KILOMETERS
  INTERVALS 1
  TRANSPARENT TRUE
  STATUS ON
END
```

Listing 8: The layers' definitions

Up to 200 raster or vector layers may be defined in any one ".map" file. Included here are just two layers, one raster and one vector. The actual data is looked for in the folder defined by the "SHAPEPATH" variable, which is, in turn, defined at the beginning of the ".map" file. For a raster dataset, a tile index shapefile is required, as defined by the variable "TILEINDEX" and for a vector dataset the shapefile is defined by the "DATA" variable. Additionally, for each "LAYER" definition:

- defines metadata as well as allowing for GML export in keeping with OGC standards.
- uses the "TEMPLATE" variable to define the ".html" format of any attribute queries.
- The "GROUP" variable is useful for grouping layers together, so that several related themes can be referred to in one operation. E.g. turning off all hydro features (rivers, lakes and streams) at once.

```
  #
  # Start of layer definitions
  #
  LAYER
    NAME drgs
    TYPE RASTER
    STATUS OFF
    OFFSITE 252 252 252
    CLASS
      NAME 'EarthSat Mosaic'
      KEYIMAGE
graphics/earthsat_keyimage.png
    END
    METADATA
      WMS_TITLE "Asia - West (150m) -
EarthSat NaturalVue Global Landsat Mosaic"
      WMS_ABSTRACT "Asia - West (150m) -
EarthSat NaturalVue Global Landsat Mosaic
represents natural color Landsat 7-derived"
      WMS_SRS "EPSG:32643"
    END
    TILEINDEX earthsat
…
…
…
  END # drgs
  LAYER
    NAME urban_areas
    GROUP cities
    TYPE POLYGON
    DATA urban_areas
    STATUS OFF
    CLASS
      NAME "Cities & Towns"
      STYLE
        COLOR 255 225 90
      END
      TEMPLATE "templates/urban_areas.html"
    END
    HEADER
"templates/urban_areas_header.html"
    FOOTER
"templates/urban_areas_footer.html"
    DUMP TRUE # allow GML export
    METADATA
      WMS_TITLE "Urban Areas"
      WMS_ABSTRACT "Outlines of Urban Areas
in the Kyrgyz Republic. (bdys. only)"
      WMS_SRS "EPSG:32643"
    END
  END # urban_area
```

Listing 9: The end of the ".map" file

```
END # Map File
```

The “.html” files:

For the above example, a number of “.html” files were involved but only a few needed to be modified for the application to work.

<u>*The initialisation file*</u>*:* - This file, called “kg_index.html” set some basic variables to tell MapServer how to work and where to fine the “.map” file. For our application only the following lines, clearly indicated by the comments in the code, needed to be modified.

Listing 10: A portion of the ".html" initialisation file

```
        // EDIT THE NEXT 2 LINES TO MATCH YOUR SETUP
        var snippet = "IMAGEPATH '/ms4w/tmp/ms_tmp/'";
        snippet += " IMAGEURL '/ms_tmp/'";
…
…
…
      <!-- EDIT THESE HIDDEN VARIABLES -->
      <input type="hidden" name="map" value="/ms4w/Apache/htdocs/Demo5/kyrgyz.map">
      <input type="hidden" name="program" value="/cgi-bin/mapserv.exe">
      <input type="hidden" name="root" value="/Demo5">
      <select name="template" size="1">
        <option value="kyrgyz_basic.html"> Basic Application
…
```

<u>*The map template file*</u>*:* - This file is simply an HTML form to display the image returned from MapServer and to interact with the user. Its name is defined in the initialisation file. The map template file needed very little modification but what was important was the values in the “Select Layers to Display” control needed to match the “LAYER” or “GROUP” names in the “.map” file

Listing 11: A portion of the ".html" map template file

```
<center><h1>MapServer - Kyrgyz Application</h1></center>
…
…
…
      <td>
      <b>Select Layers to Display: </b><br>
      <select multiple name="layer" size=6>
        <option value="airports" [airports_select]> Airports
        <option value="cities" [cities_select]> Cities
        <option value="lakes" [lakes_select]> Lakes
        <option value="rivers" [rivers_select]> Rivers
        <option value="roads" [roads_select]> Roads
        <option value="drgs" [drgs_select]> Earthsat Image
      </select>
…
…
…
<a href="[root]/kg_index.html">back to start</a><p>
<i>kyrgyz_basic.html</i>
```

The attribute template files: - Are the ".html" files needed to make up a table for the displaying of attributes. There were three files used for each table. Below are the files needed for the "Urban Areas" layer.

Listing 12: urban_areas_header.html

```
<font size+1><b>Layer: Urban Areas</b></font><p>
<table cellpadding=5 cellspacing=2 border=0>
<tr bgcolor=#CCCCCC><td
bgcolor=#ffffff>&nbsp;</td><th>Name</th><th>AREA</th><th>PERIMETER</th></tr>
```

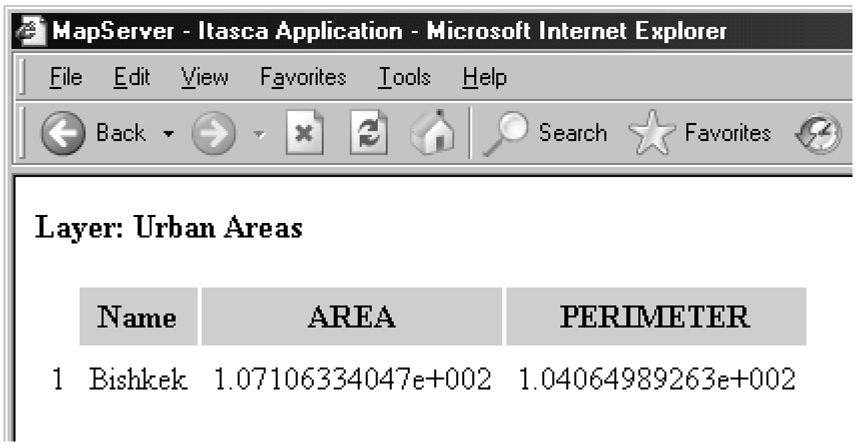
Listing 13: urban_areas.html

```
[lrn][Name][AREA][PERIMETER]
```

Listing 14: urban_areas_footer.html

```
</table><p>
```

The final result when MapServer puts these three files together is shown below in Figure 19.


MapServer - Itasca Application - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Back
Search
Favorites

Layer: Urban Areas

	Name	AREA	PERIMETER
1	Bishkek	1.07106334047e+002	1.04064989263e+002

Figure 19: A screen dump showing an attribute query example.

Appendix C – Implementation details for PostgreSQL and PostGIS

The “shp2pqsql” command will take a shapefile and convert it into a SQL command file. For example the following command:-

```
shp2pgsql -c -W CP1251 AERONAUTICAL  dcw_airports > Airports_DCW.sql
```

will convert the aeronautical shape file to the following SQL file:-

```
SET CLIENT_ENCODING TO UTF8;
BEGIN;
CREATE TABLE "dcw_airports" (gid serial PRIMARY KEY,
"objectid" int4,
"aepoint_id" int4,
"aepttype" int4,
"aeptname" varchar(40),
"aeptval" int4,
"aeptdate" varchar(254));
SELECT AddGeometryColumn('','dcw_airports','the_geom','-1','MULTIPOINT',2);
INSERT INTO "dcw_airports"
("objectid","aepoint_id","aepttype","aeptname","aeptval","aeptdate",the_geom) VALUES
('1','163','5',NULL,'2296','19850430','01040000000100000001010000007DE6821016551405B84
07C3B9C04440');
...
...
...
INSERT INTO "dcw_airports"
("objectid","aepoint_id","aepttype","aeptname","aeptval","aeptdate",the_geom) VALUES
('322','3433','5','KASHI','4445','1987 7
1','0104000000010000000101000000C786E22021015340D59FA5224EC54340');
END;
```

Listing 15: A portion of the SQL command file to load the aeronautical points to PostGIS

The options used in the shp2pgsql are:-

- **-c** automatically creates a new table using the .dbf as default column formats
- **-W** converts all attributes of the dbf are converted from the specified encoding to UTF8. This is important as the system needs to be able to handle Cyrillic as well as Latin characters[9]. Note that the resulting SQL output contains a SET CLIENT_ENCODING to UTF8 command. This is so that PostgreSQL will be able to reconvert from UTF8 to the correct encoding in the database.

9 For example the urban areas layers needs to be able to display both “Bishkek” and “Бишкек”

Once the SQL command file is created it can be loaded to the database with the following command:-

```
psql -d Kyrgyzstan_1mil -f Airports_DCW.sql -U postgres
```

in the above command “Kyrgyzstan_1mil” is the name of the database the data are being loaded to and “postgres” is the user account being used. If necessary the user will then be prompted for a password.

Alternatively, rather than sending the output of the “shp2pgsql” to a SQL command file, it is possible to redirect the output directly to PostgreSQL by using the following command:-

```
shp2pgsql -c -W CP1251 AERONAUTICAL dcw_airports | psql -d Kyrgyzstan_1mil -U postgres
```

This all means that the whole process of loading data really is relatively simple and also very quick for what is effectively an ASCII import.

The entire procedure for loading our test data set is as follows:-

```
createdb -E UTF8 -T template_postgis Kyrgyzstan_1mil -U postgres
shp2pgsql -c -W CP1251 aeronautical dcw_airports | psql -d Kyrgyzstan_1mil -U postgres
shp2pgsql -c -W CP1251 districts  ucta_kyrgyz_oblast_bdys | psql -d Kyrgyzstan_1mil
                                                                             -U postgres
shp2pgsql -c -W CP1251 lakes  dcw_lake_and_other_hydro | psql -d Kyrgyzstan_1mil
                                                                             -U postgres
shp2pgsql -c -W CP1251 rivers dcw_rivers_and_canals | psql -d Kyrgyzstan_1mil
                                                                             -U postgres
shp2pgsql -c -W CP1251 roads dcw_roads | psql -d Kyrgyzstan_1mil -U postgres
shp2pgsql -c -W CP1251 \rban_areas dcw_urban_areas | psql -d Kyrgyzstan_1mil
                                                                             -U postgres
```

Listing 16: The procedure for loading the test data to PostGIS

The final step of displaying the PostGIS data in MapServer required only three alterations to the “LAYER” definitions in the “.map” file. These were altering the DATA line and the addition of CONNECTIONTYPE command to define the type of database involved and a CONNECTION command which contains the server name, database name and username and password information. These changes are shown by the **bold text** in Listing 17 below.

```
LAYER
   CONNECTIONTYPE postgis
   NAME districts
   CONNECTION "user=ksucta password=ksucta dbname= Kyrgyzstan_1mil host=geo-server2"
   TYPE POLYGON
   STATUS DEFAULT
   DATA "the_geom from ucta_kyrgyz_oblast_bdys"
   REQUIRES "![drgs]"
   CLASSITEM 'PNAME_E'
   CLASS
     EXPRESSION 'Chu'
     STYLE
       OUTLINECOLOR 128 128 128
       COLOR 225 225 185
     END
   END
   CLASS # every other county in the state
     EXPRESSION /./
      STYLE
       OUTLINECOLOR 128 128 128
       COLOR 255 255 255
     END
   END
   METADATA
     WMS_TITLE "District/Oblast"
     WMS_ABSTRACT "Oblasts (or Districts) of the Kyrgyz republic"
     WMS_SRS "EPSG:32643"
   END
 END # districts
```

Listing 17: The changes to the ".map" file to access data in PostGIS

Appendix D – Configuring the MapServer ArcIMS Emulator

This utility proved to be very simple to install and only slightly more difficult to understand how to configure and get it running. The steps were as follows:

1. The install of the downloaded file IMSEmu-MS4W.zip (available from http://maptools.org/ms4w/index.phtml?page=downloads.html) was done by simply unzipping it over top of the c:\ms4w folder. This process installs all configuration and application files to the relevant folders.

2. A restart of Apache was done by double clicking the c:\ms4w\apache-restart.bat file.

3. The configuration file c:/ms4w/apps/imsemu-0.4/Esrimap.cfg needed to be edited with our own ".map" files added

```
# Use this file to add mapfiles
CONFIG
  TEMPDIR /ms4w/tmp/ms_tmp
  SERVICE
    NAME Example
    STATUS ON
    PROJECTION 32643
    MAP /ms4w/Apache/htdocs/Demo5/kyrgyzIMS.map
    TYPE IMAGE
  END
END
```

4. The IMS was then tested by adding an ArcIMS Server in ArcCatalog, in our test case to http://localhost

The one problem with this was, that, as only the ".map" file was used and not the ".html" files, the variables which had been defined in the ".html" file and that were needed for the ArcIMS emulator had to be redefined in the ".map" file. This was not clear from the outset and required some working through. It also meant that the variables "IMAGEPATH" and "IMAGEURL", defined in the "WEB" section of the ".map" file (see Listing 4: The WEB section on page 65 above) were now defined as:

Listing 18: Changes to the WEB section of the ".map" file to facilitate the ArcIMS emulator

```
#
# Start of web interface definition (including WMS enabling metadata)
#
WEB
  HEADER templates/header.html
  TEMPLATE "set in kg_index.html"
  FOOTER templates/footer.html
  MINSCALE 250000
  MAXSCALE 9000000
  IMAGEPATH "/ms4w/tmp/ms_tmp/"
  IMAGEURL "http://localhost/ms_tmp/"
  METADATA
    WMS_TITLE "UMN MapServer Itasca Demo - altered for Kyrgyzstan"
    WMS_ABSTRACT "This is a UMN MapServer application for Kyrgyzstan
                 located in Central Asia."
    WMS_ACCESSCONSTRAINTS "none"
    # change this value to match your setup
    WMS_ONLINERESOURCE "http://geo-server2/demo5/kg_index.html"
    WMS_SRS "EPSG:32643"
  END
END
```

Appendix E – Implementing OGC services in MapServer

This appendix describes briefly how OGC services can be implemented in MapServer by just editing the ".map" file. It summarises an extensive work assembled by Tom Kralidis for the "MapServer OGC Web Services Workshop". [Kralidis et al. 2007]

WMS – Web Mapping Service

By adding OGC parameters to the ".map" file it is possible for MapServer to respond to the WMS requests of GetCapabilities, GetFeatureInfo[10], GetMap, and DescribeLayer.

As accessing the metadata in the ".map" file is central to MapServer's ability to emulate a WMS, it is ordinarily necessary for the user to know the location of this file so it can be specified with the www address in the URL for example

```
WWWaddress/cgi-bin/mapserv.exe?map=/ms4w/apps/ksucta02/map/kyrgyzCham.map&
    version=1.1.1&service=WMS&request=GetCapabilities
```

This is a far from ideal situation for a number of reasons, e.g. the undesirable situation of exposing the directory structure to the web, and the bad practice requiring the user to know the implementation of a supposedly open GIS system. To overcome this, the documentation suggests a work around of hiding the ".map" file through the HTTP settings. This is achieved by copying and pasting the "mapserv.exe" to another filename like "wms.exe" in the "cgi-bin" folder and then referring to this new filename in the remap configuration file in the "\ms4w\httpd.d\" folder, as shown in the last line of Listing 19 below.

10 It should be noted that WMS' GetFeatureInfo is limited in that it performs only point-based queries on map data but has no ability for complex, expression-like queries, these, however, are covered the WFS specification.

Listing 19: The remap configuration file for MapServer to respond to a WMS request

```
Alias /ksucta02/ "/ms4w/apps/ksucta02/htdocs/"

<Directory "/ms4w/apps/ksucta02/htdocs">
  AllowOverride None
  Options Multiviews FollowSymLinks
  Order allow,deny
  Allow from all
</Directory>

#
# In order to hide the mapfile to OGC requests
SetEnvIfNoCase Request_URI “/cgi-bin/wms”
MS_MAPFILE=/ms4w/apps/ksucta02/map/kyrgyzCham.map
```

The new URL to request GetCapabilities from the WMS now looks like…

```
WWWaddress/cgi-bin/wms.exe?&version=1.1.1&service=WMS&request=GetCapabilities
```

…which is what a user would expect from an open GIS server.

WFS – Web Feature Service

MapServer supports basic WFS[11], and will respond to the requests of GetCapabilities, DescribeFeatureType and GetFeature. Again, like WMS, this is implemented by adding additional “wfs_*” metadata elements to the “.map” file and using the same “ows_*” elements already added previously. Most importantly for a layer to be WFS enabled its “.map” file definition must include the “DUMP TRUE” line.

Additional comments on MapServer’s implementation of OGC specifications

It is possible for MapServer to also connect as a client to other WMS servers. This is a useful feature and will be kept in mind for future developments but was considered outside the scope of this study.

Also worth noting for future development, is the fact that the SLD (Styled Layer Descriptor) is enabled WMS but it is not fully supported.

11 Basis WFS means no transactions.

Appendix F – Open source GIS functionality

Table 7 below is an abbreviation of a work carried out by Stefan Steiniger [Steiniger 2008] in which he compares the various open source GIS clients for a large variety of different functionalities. Below is listed only that which refers to

- connection to PostGIS
- language support
- OGC compatibility and
- coordinate systems support

Table 7: Showing functionality of selected open source GIS packages

	Reading Databases	Writing Databases	Database queries (SQL)	Multi language support	Languages	Supported OGC standards	Support different CoordSys
GRASS	PostGIS PostgreSQL ODBC MySQL SQLite Oracle	PostGIS (p, limited)	yes	yes	?	WMS, since GRASS 6.3 WFS, GML (via OGR)	yes
QGIS	PostGIS	PostGIS	yes	yes	26 (incl. RU)	WMS, WFS, SFS (via PostGIS) , GML (via OGR)	yes
uDig	PostGIS Oracle DB2 ArcSDE	PostGIS Oracle DB2 ArcSDE	*no*	yes	EN, FR, IT, DE, ES	WMS, WFS, WFS-T, SLD, GML, SFS, Filter	yes
JGrass	PostGIS Oracle DB2 ArcSDE	PostGIS Oracle DB2 ArcSDE	*no*	yes	EN, FR, IT, DE, ES	WMS, WFS, WFS-T, SLD, GML, SFS, Filter	yes
DivaGIS	PostGIS Oracle DB2 ArcSDE	PostGIS Oracle DB2 ArcSDE	*no*	yes	EN, FR, IT	WMS, WFS, WFS-T, SLD, GML, SFS, Filter	yes

	Reading Databases	Writing Databases	Database queries (SQL)	Multi language support	Languages	Supported OGC standards	Support different CoordSys
gvSIG	PostGIS HSQLDB MySQL Oracle-Spatial	PostGIS Oracle-Spatial	?	yes	N, ES, VA, GL, CZ, DE, EU, FR, IT, PT, CN, PL, RO	WMS (1.1, 1.1.1, 1.3.0), WFS, WCS (1.0, 1.0.0), WCS (2.0), Filter (1.0)	yes
SAGA	*no (intended)*	*no (intended)*	*No (intended)*	yes	EN, DE	-	yes
JUMP	PostGIS Oracle(p) ArcSDE(p)	PostGIS (p, limited)	*no*	*no*	-	SFS, WMS, GML, WFS(p)	yes(p)
OpenJUMP	PostGIS Oracle(p) ArcSDE(p)	PostGIS (p, limited)	*no*	yes	EN, Fi, DE, FR, ES, P, IT	SFS, WMS, GML, WFS(p)	yes(p)
SkyJUMP	PostGIS Oracle(p) ArcSDE(p)	PostGIS (p, limited)	*no*	*no*	-	SFS, WMS, GML, WFS(p)	UTM-Geo yes(p)
Pirol JUMP	PostGIS Oracle(p) ArcSDE(p)	PostGIS (p, limited)	*no*	yes	EN, DE	SFS, WMS, GML, WFS(p)	yes(p limited)
DeeJUMP	PostGIS Oracle(p) ArcSDE(p)	PostGIS (p, limited)	*no*	yes	EN, Fi, DE, FR, ES, P, IT	SFS, WMS, GML, WFS(p)	yes(p)
ILWIS	*no*	*no*	*no*	?	?	under develop-ment WFS, WCS	yes
KOSMO	PostGIS Oracle-Spatial/ Locator MySQL	MySQL Oracle Postgres	*no*	yes	EN, ES, PT, RU	WMS, SFS, SLD (under developmen t WFS, WPS, GML)	yes
MapWindo GIS	PostGIS(p)	PostGIS(p) MDB(p)	*no*	yes	CS, DE, EL, EN, ES, FA, FR. IT, JP, NL, PT, TH, ZH-CHS/ZH-CN	WMS (P), WFS (P)	yes