MSC INFORMATION SYSTEMS
SCHOOL OF COMPUTING AND INFORMATICS
UNIVERSITY OF NAIROBI

CONSERVATION WEB-GIS APPLICATION OF THE VIRUNGA NATIONAL PARK, DR CONGO USING OPEN SOURCE GIS TECHNOLOGIES

P56/7519/2006

Maritim Zachary Kimutai

11/04/2013

Supervisor: Mr Andrew Mwaura

A thesis submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

INFORMATION SYSTEMS

DECLARATION

This thesis, written by Maritim Zachary Kimutai, and entitled Conservation Web-GIS Application using Open Source GIS Technologies of the Virunga National Park, Democratic Republic of the Congo, having been approved in respect to style and intellectual content, is referred to you for judgment.

This project has been submitted for examination with my approval as University supervisor:

_____

Andrew Mwaura, Supervisor

I declare that this systems project is my original work and has not been submitted for a degree in any other university:

_____

Maritim Zachary Kimutai

Date of Defence: April 11, 2013

University of Nairobi, 2013

DEDICATION

I dedicate this systems project first and foremost to God, to my family for their sacrifice in support of my studies, to the memories of my late father Stephen Kitur and to WWF in her unrelenting efforts to create a world in which humans and nature will live in harmony.

ACKNOLWEDGEMENTS

I wish to express my sincere thanks to my supervisor Andrew Mwaura for his guidance, support, encouragement and positive criticism while undertaking this project.

My gratitude also goes to Michael Ngugi of Regional Centre for Mapping of Resources for Development (RCMRD) for the many aspects of Web-GIS development that we discussed together. The same goes out to Mr. Moturi for taking me through Database Design Matters.

Further gratitude extends to my current employer (WWF ESARPO) for the unwavering financial support and time allocation to pursue this worthy course.

This project has benefited immensely from the excellent inputs of these dedicated people and institution. I hope I have been able to put together some of their wonderful contributions and of others not mentioned.

ABSTRACT

The availability of conservation data and information of a conservation area (such as a national park) can lead to an improvement in the exploitation of its natural resources, management and preservation and to a greater fruition by the local and international community. To this aim, a Web-GIS of Virunga National Park (*Parc National des Virunga, PNVi*) - a natural and a UNEP world heritage site and among the oldest parks in Africa located in the eastern part of Democratic Republic of the Congo (DRC) – was designed and implemented. The main goal was to create a tool, easy to access and suitable both for domain experts, such as conservationists or museum curators, and laypersons interested in the cultural assets of the area for educational or tourist purposes. Territorial and environmental information was organized in two different parts, in order to provide a complete and exhaustive frame where the environmental protection entities are located. The first one allows the analysis of the environmental and administrative aspects of the area, providing several groups of thematic maps; the second one deals with the study of the conservation features (distribution/occurrence of species), giving their position and their main characteristics (such as taxonomic classification) and features.

Data sources were both digital data (existing and *in-situ* acquired) and non-digital hard copy maps which were scanned, georeferenced and digitized during the data mobilization phase. All data was processed and loaded in the system as vector (shapefiles) and raster (GeoTIFF files) which were then subsequently stored in PostgreSQL DBMS tables. The whole system lives in a common web site, implemented in Python, HTML and Javascript languages and exploiting GeoServer for GIS functionalities and PostGIS (spatial data engine) for its connection with the PostgreSQL database as well as GeoNode for its map-rendering/viewing and creation features.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

ArcSDE - ArcGIS Spatial Data Engine for rendering maps in an interface.

COTS – Commercial off the Shelf Software

DB - Database

DRC – Democratic Republic of the Congo

ESARPO – Eastern and Southern Programme Office of the WWF

ESRI – Environmental Systems Research Institute, Redlands California, USA

GIS - Geographic Information Systems

GNU – General Public License for open Source Systems

GPS - Global Positioning System, a satellite-based positioning system

ICCN- *Institut Congolais pour la Conservation de la Nature*, Congolese Institute for Nature Conservation

JDBC - Java Database Connectivity

NGOs – Non-Governmental Organizations

OGC - Open Geospatial Consortium

ORDBMS – Object-Relational Database Management System

PNVi- *Parc National des Virunga*, Virunga National Park

Q-GIS – Quantum Geographic Information System, Open Source Software

RAM – Random Access Memory

SDLC – System Development Life Cycle

SQL – Structured Query Language

TanBIF - Tanzania National Biodiversity Information Facility

UCL – *Université Catholique de Louvain*, Louvain Catholic University in Belgium

URL – Universal Resource Locater

UTM – Universal Transverse Mercator, a Coordinates Map Projection System

WFS - Web Feature Service

WKB - Well-Known Binary

WKT - Well-Known Text

WMS - Web Map Service

WWF – World Wide Fund, A global conservation organization

1.0 INTRODUCTION

1.1 BACKGROUND

As the Democratic Republic of the Congo emerges from its first democratic elections since independence and as the whole world begins to develop confidence in the future of this remarkable country, the Congolese people will have to take stock of its numerous resources, abundant and precious, human and natural, in order to begin rebuilding their country after the most bloody civil wars in history. Top in the list shall be the national parks, the pillars of the emerging tourist industry and a crucial heritage for the future generations. Then all the eyes will turn towards Virunga National Park, the most precious of the Congolese parks, the emblem, not only for Congo but for the entire African continent. This park cannot be described but by the superlatives: the oldest park in Africa, the most threatened; home for the greatest number of mammals, great apes, birds and reptiles; the diversity of its habitats and its landscapes that is unequaled, and the list is endless (Languy M. & E. deMerode, 2006:13). Certainly the difficulties experienced during the past decades have caused enormous destruction to the park but its biological and esthetic value remains and it's evident that this park can be restored. From its situation today, it can be said that Virunga National Park survived the many difficulties but this was not by chance or by luck but it was through the efforts of the Congolese Institute for Nature Conservation (ICCN) and some conservation NGOs (including WWF).

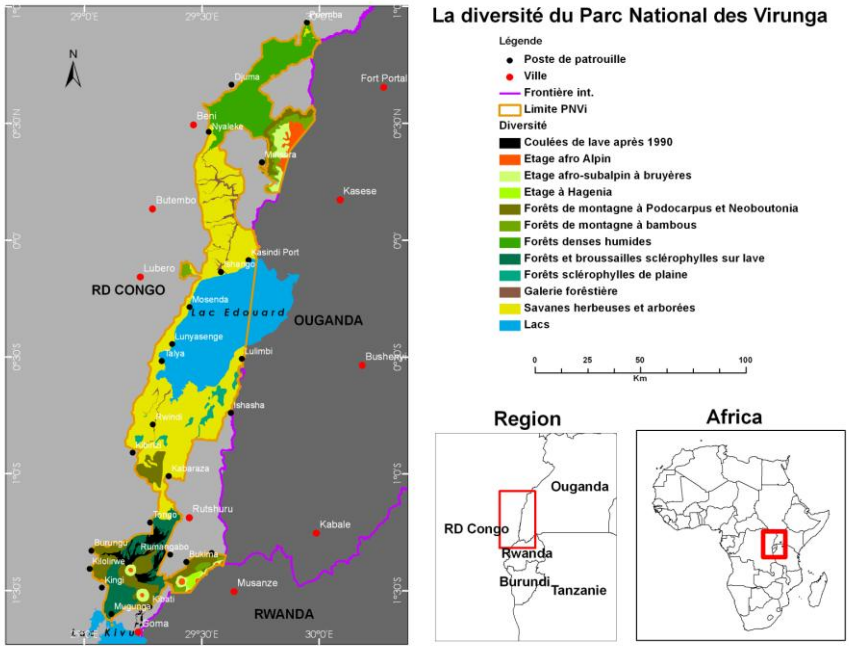

Figure 1: Location Map and Virunga National Park's diversity

1.2 JUSTIFICATION

This project aims at developing an open source Web-GIS application through analysis and design of a Geographic Information System (GIS) database that will go a long way in contributing towards a more scientific and efficient management of Virunga National Park. Evidently, being the oldest park in Africa, there has been a lot of documentation of works on it but these efforts have not been integrated into a system that provides for efficiency and time series aspects for which statistical analyses that will help in decision-making and intervention response can be made. The various types of information that will be disseminated through the web application will include, but will not be limited to: participatory demarcation of the park boundary to avoid conflicts, biodiversity studies, socio-economic surveys, vegetation mapping, etc. An Enterprise Geographic Information System (GIS) will integrate all spatial and non-spatial attribute information in an Enterprise Geodatabase (under PostgreSQL/PostGIS) that will guide conservationists in their efforts to save Virunga National Park. A Geodatabase is a relational database that stores geographic data and is a container for storing spatial and attributes data and the relationships that exist among them. This Enterprise Geodatabase:

- Provides a Uniform Repository for Geographic (Vector and Raster) and attribute data that is scalable.
- Provides more efficient data entry and editing through rule and constraint application.
- Provides for Multi-user editing capabilities through PostGIS technology with versioning in a centralized DB environment.
- Enables easy extraction/mining of data through DB views and/or direct SQL querying + modeling via the intuitive Web Interface.

1.3 PROBLEM DEFINITION

Geomatics is the combined use of the technologies of Remote Sensing via satellite imaging, Global Positioning Systems (GPS) and digital mapping via Geographic Information Systems (GIS). These have become essential tools for delimitation, demarcation, surveillance and, increasingly, the management of protected areas. Protected Areas in sub-Saharan Africa are increasingly under pressure from growing human populations, posing a complex challenge for their continued conservation ((Languy, M. & deMerode, E. 2006:289). Virunga National Park authorities are facing a growth in the illegal settlement of populations within the park. As a result, the authorities have begun to dialogue with all concerned parties, including the populations themselves to explore the possibilities of shifting these populations to areas outside

the park. The major challenge is to identify those settlement sites in order to come up with the best forms of interventions to evacuate the populations voluntarily. GIS will provide maps that will aid in planning such operations and interventions. Unlike traditional paper maps, GIS will provide an environment for continuously updating and editing the maps as new information comes to light. To do these, GIS software must be made available for installation and use. Such GIS software e.g. ArcGIS, MapInfo, etc are only available through expensive licensing agreements, which in turn limits the use of the technology. This is the first time in the region that Open Source GIS infrastructure is being applied to bridge the gap existing in budgetary constraints and to enable the stakeholders in conservation to allocate these budgets to other crucial tasks such as building hospitals and clinics for the populations that are being evacuated from the park.

1.4 GOALS AND OBJECTIVES

The goal of this research is to create (using Open-Source Technologies) a web-based GIS tool, easy to access and suitable both for domain experts, such as conservationists or museum curators, and laypersons interested in the cultural assets of Virunga National Park for conservation, educational or tourist purposes.

The specific objectives of this study are:

1. To identify and evaluate existing data and conservation strategic plans of *Parc National des Virunga* of the *République Démocratique du Congo (*DRC*).*
2. To design, develop, configure and tune an enterprise spatial & attribute database (under PostgreSQL /PostGIS).
3. To collect, collate, process, clean and migrate existing flat-file attribute and vector data in MS Excel, MS Access and Shapefiles as well as raster data such as SPOT, Landsat, Aerial Photos & ASTER formats into the PostGIS Geodatabase.
4. To develop, test & operate a Web-GIS Application for spatial data and information exchange.

2.0 LITTERATURE REVIEW

2.1 REVIEW OF THE VIRUNGA NATIONAL PARK

2.1.1 The Past

This covers the creation of the first park in Africa and the efforts which were needed to enable its maintenance during Congo's turbulent past. This period is subject to a lot of controversies. Analysing the events of the past brings to light the various different elements both legal and social that characterized the establishment of the first African park together with its ulterior expansion. The Virunga is today one of the richest regions in terms of literature. In the distant past, it became the heart of a long turbulent history to which is traced the first appearance of man dating to 18,000 to 13,000 BC around Lake Edward and Ishango. In brief, the past looks at the following various elements of the park:

- Lake Albert National Park: The birth of the first park in Africa (1925-1960).
- Life in Albert National Park between 1925 and 1960.
- The rebirth of the National Park (1960-1991)
- The years of the crisis (post 1991)
- Evolution and Application of scientific research in Albert National Park/Virunga National Park.
- Eighty years of volcanic activities in the park and their ecologic and socio-economic impacts.
- Eighty years of vegetation dynamics.
- Evolution of the populations of the great mammals.
- The transition of the Landcover/Landuse systems around the park.
- Development of tourism in DRC and in the great lakes region at large.

2.1.2 The Present

Here, the current status of the population of the great vertebrates is brought to light. The fishing problems and challenges of controlling them are looked at and the wood fuel crisis is discussed. In addition, the management capacity of the Congolese Institute for Nature Conservation (ICCN) is analysed and recommendations made. It is worth noting that the

political challenges and difficulties encountered in North Kivu province play a very crucial role in determining the future of conservation of the Virunga National park. The implication of the local communities in the management of the park is also a very important part in management efforts. A framework for crisis management needs to be put in place. Finally, the recent springing up of threats to the existence of the park needs to be addressed while drawing lessons from the 80 years of management efforts in this park.

2.1.3 The Future

The future of Virunga National Park is what has motivated this project. Looking at the past and the present of this park, it is necessary to explore the approaches and the tools available in order to assure its survival. It is in the light of this that a system for surveillance and monitoring of the park including continued research remains very important. Such a system would be key to the restoration and future management of the Virunga. The system will be made up of spatial sciences, comprising of Telemetry, Remote Sensing and Geographic Information System (GIS) that will form an integral part in the day-to-day running of the park's activities and in the long term planning of future activities as well as in decision-making efforts. The system will be scalable hence taking into account enormous data and information collection and temporal (time series) issues.

2.2 REVIEW OF OTHER OPEN DATA AND OPEN SOURCE SYTEMS

2.2.1 Existing Works in the Area

Open Data systems are now coming of age and have arisen as a result of the realization that we are in the information age where information is power and that for human development to be achieved information sharing is paramount. Open source systems deliver quality systems to humanity and such systems are available as community tools that global communities can participate in their improvement. These systems also save organizations the huge costs of commercial off-the shelf applications (COTS). Some of the existing works in the open source systems world, and from which this project has variously borrowed from include:

- The Kenya Government Open Data system where government information and data are shared on the government portal and is freely accessible to citizens and whoever is interested. Shared information includes census data (and their analysis); school mapping data; health centres and many more (https://opendata.go.ke/). The portal however stops at just sharing data and information but does not provide a platform for

value addition through map-making and hence certain graphical representations for easier decision-making are lacking.

- The GeoNode Project that handles participatory open source cartography for map creation, editing, publishing as well as data sharing. GeoNode is a platform for the management and publication of geospatial data. It brings together mature and stable open-source software projects under a consistent and easy-to-use interface allowing users, with little training, to quickly and easily share data and create interactive maps. GeoNode provides a cost-effective and scalable tool for developing information management systems (http://geonode.org/)
- Tanzania's Biodiversity Information Facility (TanBIF) which handles both taxonomic and occurrence biodiversity data implemented as a bioinformatics project with spatial information stored in the open source PostGIS/PostgreSQL platform. TanBIF is the Tanzania's National Biodiversity Information Facility, an extensive, decentralized system of national biodiversity information units that intends to provide free and universal access to data and information regarding Tanzania's biodiversity (http://www.tanbif.or.tz/). The system is implemented using Quantum GIS (Q-GIS) which is an open source GIS platform.
- Works by *Institut Congolais pour la Conservation de la Nature* (ICCN). This Congolese institute for nature conservation has done a lot of GIS data collection for species and biodiversity monitoring and was one of the major sources of the GIS information used in this study. Their work however stopped and flat-file level and we hope that this system under development will provide ICCN with the much-needed structured database.

2.2.2 PostgreSQL & PostGIS

2.2.2.1 What is PostgreSQL?

PostgreSQL is an object-relational database management system (ORDBMS) based on POSTGRES, Version 4.2, developed at the University of California at Berkeley Computer Science Department. POSTGRES pioneered many concepts that only became available in some commercial database systems much later.

PostgreSQL is an open-source descendant of this original Berkeley code. It supports a large part of the SQL standard and offers many modern features:

- complex queries

- foreign keys
- triggers
- views
- transactional integrity
- multiversion concurrency control

Also, PostgreSQL can be extended by the user in many ways, for example by adding new

- data types
- functions
- operators
- aggregate functions
- index methods
- procedural languages

And because of the liberal license, PostgreSQL can be used, modified, and distributed by anyone free of charge for any purpose, be it private, commercial, or academic.

2.2.2.2 What is PostGIS?

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 backend 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.

PostGIS development was started by Refractions Research as a project in open source spatial database technology. PostGIS is released under the GNU General Public License. PostGIS continues to be developed by a group of contributors led by a Project Steering Committee and new features continue to be added.

The OGC WKT (Well-Known Text) and WKB (Well-Known Binary) form define type and coordinates of an object. Data from a PostgreSQL/PostGIS database can be used as data source for spatial server software like MapServer and GeoServer. PostGIS is licensed under the GNU GPL and operated as a Free Software Project (http://postgis.refractions.net/)

2.2.2.3 PostGIS Spatial Data Management

PostGIS is implemented compliant to the OGC Simple Feature Specifications for SQL standard. The OGC specification defines operations and the SQL schema to insert, query, manipulate and delete spatial objects. The coordinates of the spatial objects are stored in Feature Tables. One Feature Table can contain one type of geometry (point, line, polygon, multiple of each and geometry collection). The coordinates of each object is stored in a field of a special type. The field type for a set of coordinates is WKT (Well Known Text). Meta data is collected for each Feature Table to organize the type and the coordinate system of the contained geometry. The metadata for each Feature Table is stored in the special table geometry_columns. PostGIS ships with a Shapefile-loader and dumper. Various file types (Shapefile, MapInfo, DGN, GML, etc) can be read, converted and inserted to a PostGIS database using the OGC libraries. A PostGIS Feature Table can be used as data source for a growing variety of map and feature server software like UMN MapServer, GeoServer, uDGI, deegree, JUMP, etc. The data can be accessed using standard ODGB or JDBC connectors. Several projects have evolved around PostGIS transforming and inserting geometries from highly specialized formats like SICAD C60, EDBS, DXF, WLDGE and many more.

3.0 RESEARCH DESIGN AND METHODOLOGY

3.1 THE SOLUTION

3.1.1 Background

The research design used is that of systems/applications development involving Open Source Technologies in addressing Conservation Non-Governmental Organizations' (NGOs') geographic information data needs. The solution adopted was a web-based GIS system for Virunga National Park in which an interactive customized map application that runs on any browser, multiple computer platforms and operating systems was developed and implemented. In performing this function, the Web-GIS was designed using a server - client architecture consisting of three well-defined and separate processes, each running on a different platform:

- The user interface, which runs on the user's computer (the client).
- The functional modules that actually process data. This middle tier runs on a server and is often called the application server. In our case the application server is the GeoServer.
- A database management system (DBMS) that stores the data required by the middle tier. This tier runs on a database server which in our case is PostgreSQL/PostGIS server.

The three-tier design has many advantages over traditional two-tier or single-tier designs, the chief ones being:

- The added modularity makes it easier to modify or replace one tier without affecting the other tiers.
- Separating the application functions from the database functions makes it easier to implement load balancing.

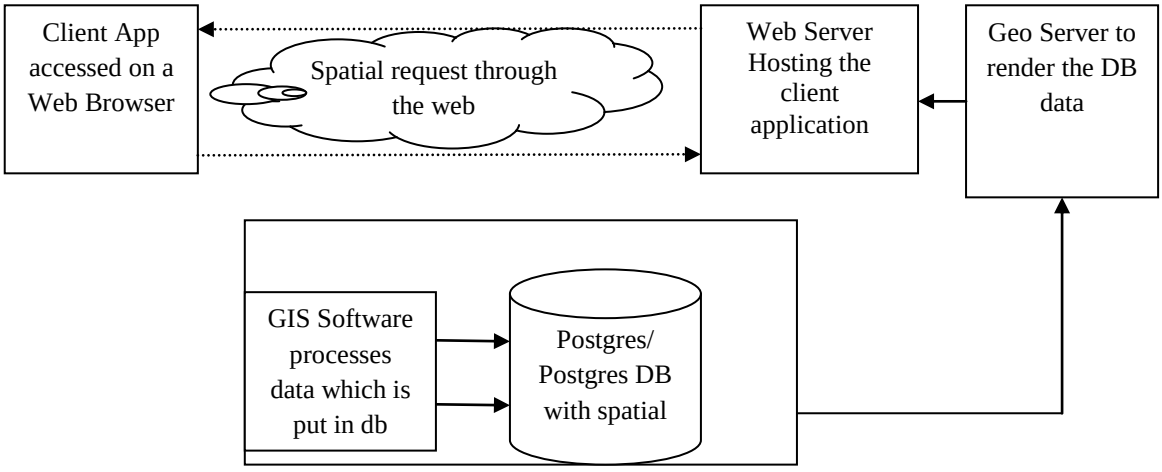

Figure 2: Web based GIS system Architectural design Layout

The above named architecture widely exists to handle large volumes of data across many users. Some computers act as servers and others act as clients. A Server simply has a web server (Apache) and database server (Postgres/PostGIS) running, on the client side, a customized interactive map application will be running and thirdly a server (GeoServer) that renders the spatial and non-spatial data produced by the proprietary GIS software. The GeoServer runs as a middleware at the server side to communicate between the client and the server.

3.1.2 Server - Client Architecture (Server Side Applications)

In developing the web-based GIS system, we therefore implemented the Server-Client architecture. In this kind of architecture, the clients only have an interactive map-user interface to communicate with the server and display the results. All the processing is done on the server. This enables the application to speed up the execution of spatial data in terms of capturing, storing, analyzing and presenting spatial data. The server computer usually has more power than the client, and thus manage the centralized resources (i.e. spatial data).The main functionality is on the Server, where it is possible to add any utility programs that can be linked to the server software incase such a need is identified. Major advantages of this model driven by Database centralization are:

- Central control
- Easy for data eminence/updating
- Keep the latest version
- Generally cheaper
- Integration possibilities
- Easier manipulation of cartographic aspects such as font, symbology etc

3.2 THE OPEN SOURCE WEB-GIS BUILDING STEPS

To design, implement and test PNVi-WEB GIS, the following steps were taken:

1. Territory knowledge and data acquisition;
2. Data pre-processing, processing
3. Database design, software development and interface design;
4. Web GIS implementation, test and data loading and sharing;

3.2.1 Territorial knowledge and data acquisition

The first step underlines some data availability issues, which can be grouped into the generic categories of data lack, data incompleteness and data non-uniformity. To analyse and deal with this situation some direct territorial inspections were made. My several years of working

with various organizations and local people in the region made it easier to mobilize data. The universities of Ghent and Louvain in Belgium also contained numerous types of information in scanned maps and aerial photographs which were sourced and subjected to the next phase of analysis.

3.2.2 Data pre-processing and processing

Once the available data and their formats and characteristics were known, the second step involved pre-processing the data. Here the old topographic map sheets and aerial photos were straightened up and prepared for scanning and subsequent digitization. The scanner shown below (Figure 3) was used to carry out the scanning exercise. Thereafter, the scanned maps were cropped, geo-referenced and then digitized. All data was processed for structuring the information they bring and converted in those formats which can be supported by the developed application.

Figure 3: Océ CS4236 Scanner (Large Format)

3.2.3 Database design and interface development

This was the third step used in the implementation, that of database design and the development of the applications interface for interacting with those data and database. In this case "interacting" means not only how to load and manipulate data – read DBMS - but also how to display them in the Web-GIS with a correct georeferencing – read GIS engine and spatial enabling support. So, this step coincided with two parallel operations, data processing and applications development. The application was set and implemented for its spatial

functions, mutual interaction and web utilisation. The interface was coded in python and uses any web browser.

3.2.4 Web-GIS implementation, testing and data uploads and sharing

Afterwards, all data calls were loaded in their specific setting files and application was connected and inserted in an expressly implemented common web site. In this way the web GIS was completed. The last step involved testing the Web-GIS workings, efficiency and effectiveness and consequently to improve data processing, application tuning and implementation and web site pages design.

3.3 SYSTEM DEVELOPMENT CYCLE (SDLC)

To provide successful implementation to meet the requirement of the needed Web-based GIS system, the figure 4 below shows the Web-GIS system development cycle that we used, which is described in terms of 7 major activities starting with the requirements analysis and ending with on-going use and maintenance of the Web-GIS system.

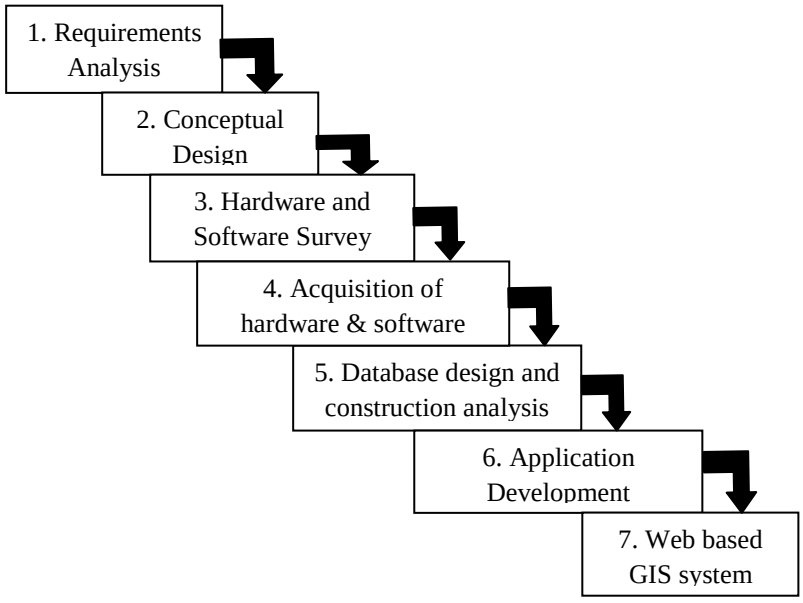

Figure 4: Web Based GIS system Development Cycle

3.3.1 Requirements Collection and Analysis

The requirement analysis step was performed through discussions with potential users to identify the needs of all stakeholders: Some of the things that were discussed during the requirements collection and analysis include:-

- A list of the required GIS functions needed in the application. The required functions are the basic visualization functions such as Pan, Zoom, Upload, download tasks and more advanced functions such as object identification (metadata). Clients can use these functions to view the various types of information on an interactive map.
- A master list of available/needed geographic data and any other GIS layers needed as well as their related attributes.
- Compilation and review of the available data.
- Identify the data gaps.
- Data capture, i.e. mobilization of hard-copy data to generate digital data.

The information gained in the requirement analysis activity directly went into the Conceptual GIS Design activity. The requirements were collected and analysed according to the cycle of events depicted in the diagram below (Figure 5).

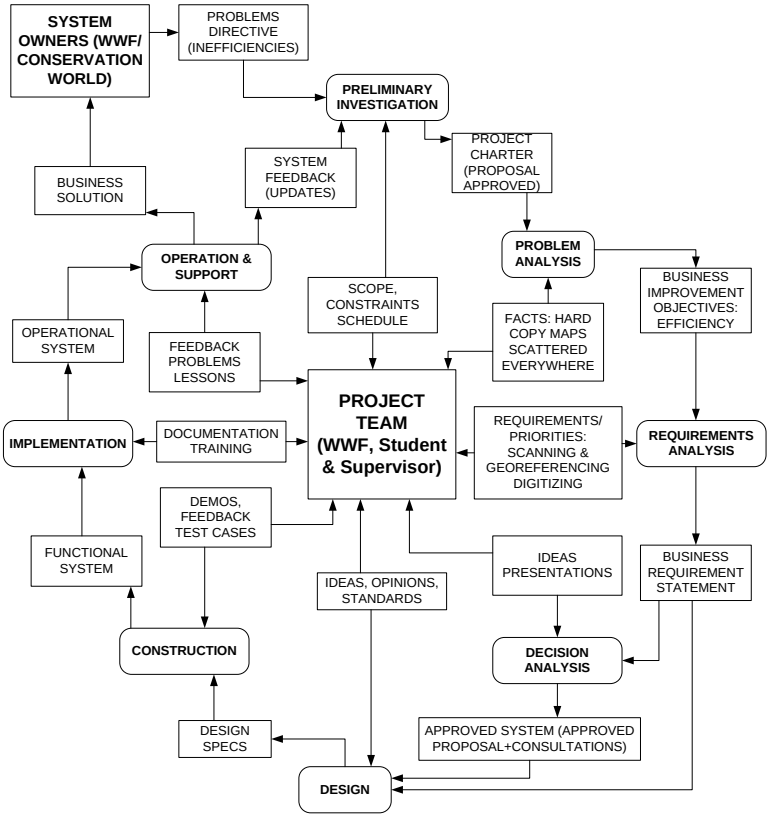

Figure 5: Requirements Collection Feeding into System Development Cycle

3.3.2 Findings from the Requirements Analysis

The requirements collection involved discussing with the WWF management and potential users of the system the preferred ways for knowledge management and data storage as well as retrieval. The following were findings from the discussions and analysis of requirements:

- That the system should be web-based and accessible to field staff in different countries through internet connectivity
- That there should be a level of security to the system and authentication of users.
- That there should be training of the users in data cleaning before uploading to the system
- That the system should enable simple map-making by novice users so as to allow for decisions to be made based on the maps made.
- That the system should be centralized and managed administratively from the WWF head offices in Nairobi with a GIS Manager in charge of gate-keeping and data quality control. This should be the system's super-user.
- That hard copy paper maps need to be scanned and georeferenced so that they can be discoverable for use in change analysis both for habitats and livelihoods.
- That the interface should be as simple as possible with minimum functionalities since the principal objectives are to store, share and manage GIS data. Complex analyses can be done with other software as long as data can be easily found online.
- The park's banner needs to be displayed on the home-page as theme.
- That GIS should be institutionalized and not left in the hands of a few people.
- That the system should be interoperable with other existing online mapping systems such as Google Maps.
- That hardware and relevant server software be purchased and configured.

3.3.3 Conceptual Design

Once the requirements were collected and analysed and required data identified, the data model that identifies the entities and their relationships was designed. The data is delivered through a central server, and clients have access to raster and vector formats in form of an interactive map that runs on an Apache Web server.

3.3.4 Hardware Survey and Acquisition

The hardware requirements were specified for the development of the system. The following minimum specifications for hardware were recommended:

- For the server to host the system:

- **Form factor**: Dependent on whether there is an existing mounting rack, else a Micro ATX Tower will be sufficient.
- **Processor:** Quad-Core Intel Xeon processor E5405 2.00GHz.
- **Front side bus**: 1333MHz.
- **Memory**: 2GB Memory (2x1GB), 667MHz Single Ranked DIMMs.

- And for the laptops/desktops to aid in system developing and eventually as client computers:
  - Minimum 2GB of RAM
  - Minimum 40GB of storage
  - About 2.4GHz of processor speed
  - External backup disks were also recommended to data security.

Taking into consideration the fact that nature conservation organizations always have limited budgets and that purchasing commercial-off-the-shelf (COTS) applications would be costly and unsustainable in the long run, it was decided that Open Source platforms should be explored to meet most of the software requirements for implementing and operating a Web-GIS. This is how we settled for PostgreSQL DBMS, GeoServer web-server and Quantum GIS (QGIS) for data preparation before uploading.

3.3.5 Database Design and Construction

The primary purpose of this phase of the Web-GIS development process was to specify "how" the Web-GIS will perform the required applications processing. Database design involved defining how graphics will be symbolized (i.e., color, weight, size, symbols, etc.), how graphics files will be structured, how non graphic attribute files will be structured, what is the active layer, in what scale shall the layers expose, how GIS products will be presented e.g. data formats, and what management and security restrictions will be imposed on file access. Other activities to be undertaken are:

- Selecting the data source (document, map, digital file, etc) for each entity and attribute included in the Entity-
- Relationship diagram
- Setting-up the actual database design (logical/physical design)
- Defining the procedures for converting data from source media to the database. Since the formats of the data needs to be compatible with each other.
- Define procedures for managing and maintaining the database.

3.4 SETTING THE ENVIRONMENT

Once the sources of all data were known and all data cleaned and pre-processed and stored, it is now possible to see the web GIS structure and how this information is used in it. PNVi-GIS web GIS is formed by five different component programs:

- GeoServer Web-GIS engine;
- Apache Web Server;
- A browser,
- PostgreSQL DBMS,
- PostGIS DB spatially enabling support.

Indeed to implement a web GIS only the GIS engine, the web server and the browser are enough; in fact, PostgreSQL and PostGIS are used because some data are stored in the system as DB tables and not as simple vector or raster files which can be directly loaded by the GIS engine.

3.4.1 GeoServer and the Web-GIS core

GeoServer is an open source program created and developed by the Open Planning Project (TOPP). It will be the GIS engine of Web-GIS: it acquires and processes requests coming from the user and returns him/her the output results. GeoServer consists of three different components:

- The map file,
- The template files,
- The CGI program.

The map file needs to set cartographic parameters, cartographic objects, data loading, classification, displaying and querying and graphic elements definition and use. It is implemented using GeoServer software's built-in object oriented scripting language with which it is possible to design how to create and use the maps and their layers. In particular, in the map file Layer Objects the paths and connection types to data load are specified: there is a direct connection for shapefiles and raster files while more complex connections are necessary for other data file formats; among them, OGR connection (by OGR library) and PostGIS connection (by PostGIS program) are used in PNVi-GIS respectively for MapInfo vector file and PostgreSQL tables.

The template file is a common HTML page provided with GeoServer specific parameters and variables. The template files are the files the user see by his browser, so they are implemented to present maps, cartographic objects, query and all other information which the web GIS designer want to offer to the user. The CGI program is the real engine of the web GIS: started up by the web server,

it reads and processes both the map file settings and the template file user defined parameters or variables and returns the processed outputs as maps, cartographic objects, variables values and query results shown in the template files. Every CGI output is a temporary image or value updated at each CGI work session.

GeoServer is a Java-based software server that allows users to view and edit geospatial data. Using open standards set forth by the Open Geospatial Consortium (OGC), GeoServer allows for great flexibility in map creation and data sharing.

3.4.2 Open and Share Your Spatial Data

GeoServer allows you to display your spatial information to the world. Implementing the Web Map Service (WMS) standard, GeoServer can create maps in a variety of output formats. OpenLayers, a free mapping library, is integrated into GeoServer, making map generation quick and easy. GeoServer is built on GeoTools, an open source Java GIS toolkit.

There is much more to GeoServer than nicely styled maps, though. GeoServer also conforms to the Web Feature Service (WFS) standard, which permits the actual sharing and editing of the data that is used to generate the maps. Others can incorporate your data into their websites and applications, freeing your data and permitting greater transparency.

3.4.3 Use Free and Open Source Software

GeoServer is free software. This significantly lowers the financial barrier to entry when compared to traditional GIS products. In addition, not only is it available free of charge, it is also open source. Bug fixes and feature improvements in open source software are greatly accelerated when compared to traditional software solutions. Leveraging GeoServer in your organization also prevents software lock-in, saving costly support contracts down the road.

3.4.4 Integrate With Existing Mapping APIs

GeoServer can display data on any of the popular mapping applications such as Google Maps, Google Earth, Yahoo Maps, and Microsoft Virtual Earth. In addition, GeoServer can connect with traditional GIS architectures such as ESRI ArcGIS.

3.4.5 GeoServer Integration Capabilities

OpenLayers makes it easy to put a dynamic map in any web page. It can display map tiles and markers loaded from any source. OpenLayers has been developed to further the use of geographic information of all kinds. OpenLayers is completely free. Google Maps (formerly Google Local) is a web mapping service application and technology provided by Google, that powers many map-based services, including the Google Maps website, Google Ride Finder, Google Transit,[1] and maps

embedded on third-party websites via the Google Maps API.[2] It offers street maps, a route planner for traveling by foot, car, bike (beta), kayak,[3] or public transport and an urban business locator for numerous countries around the world. Google Maps satellite images are not updated in real time; they are several months or years old.

Bing Maps (previously Live Search Maps, Windows Live Maps, Windows Live Local, and MSN Virtual Earth) is a web mapping service provided as a part of Microsoft's Bing suite of search engines and powered by the Bing Maps for Enterprise framework.

3.4.6 Goals

GeoServer aims to operate as a node within a free and open Spatial Data Infrastructure. Just as the Apache HTTP Server has offered a free and open web server to publish HTML, GeoServer aims to do the same for geospatial data.

3.4.7 Features

GeoServer reads a variety of data formats, including:

- PostGIS
- Oracle Spatial
- ArcSDE
- DB2
- MySQL
- Shapefiles
- GeoTIFF
- GTOPO30
- ECW, MrSID
- JPEG2000

Through standard protocols it produces KML, GML, Shapefile, GeoRSS, PDF, GeoJSON, JPEG, GIF, SVG, PNG and more. In addition, one can edit data via the WFS transactional profile (WFS-T). GeoServer includes an integrated OpenLayers client for previewing data layers.

GeoServer additionally supports efficient publishing of geospatial data to Google Earth through the use of network links, using KML. Advanced features for Google Earth output include templates for customized pop-ups, time and height visualizations, and "super-overlays".

3.4.8 PostgreSQL, PostGIS and their interaction with GeoServer

We saw that GEODATABASE feature datasets and feature classes are stored as PostgreSQL tables; in this way PostgreSQL becomes an indispensable system component from which the web GIS loads data to be displayed in the maps; we said also that these tables are called by GeoServer using the map file PostGIS connection. But obviously this loading couldn't be possible if the tables are not georeferenced. In fact, each PostgreSQL table has been previously provided with a Geometry Column, in which every record has its spatial description by a pair of UTM North, East coordinates for each point forming the feature stored in the record; in this way the tables become "spatial tables". The Geometry Column provided spatial information will be done by PostGIS: using the special AddGeometryColumn function for the archaeological remains table and automatically with PostGIS Data Loader for paths and highlights tables. So, for loading spatial tables it is enough to specify in a map file Layer Object:

- The PostGIS connection type;
- The connection parameters, in particular the name of the database containing the spatial table to be loaded;
- The name of the spatial table and its Geometry Column;
- The loading filter with the syntax used for a SQL query WHERE clause.

In this way GeoServer accesses PostGIS/PostgreSQL data like any other PostgreSQL client and it can display PostgreSQL table features using PostGIS as spatial enabling support. In figure 6 we can see an example of using PostgreSQL data with GeoServer.

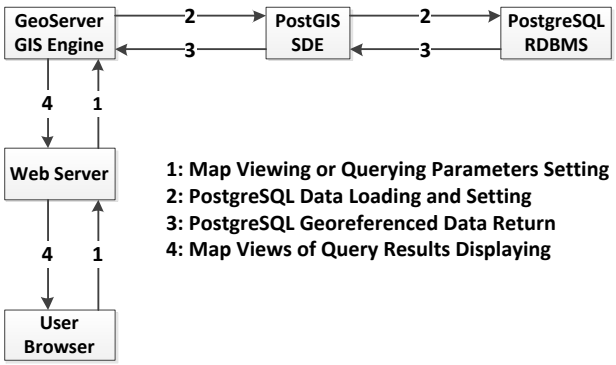

Figure 6: Using PostGIS/PostgreSQL data with GeoServer.

The possibility to use PostgreSQL data in GeoServer is available since GeoServer 3.5 release: it's quite clear that this improvement is very powerful when we need to load GIS frequently updated data in a web. For PNVi Web-GIS that means to use web GIS functionalities on very dynamic data like GEODATABASE. In fact it's more useful and conceptually correct to update a DBMS table than a

shapefile! Thus, coupling GeoServer to PostGIS/PostgreSQL, an archaeologist which has the necessary database privilege can, for example, add a newly discovered archaeological find or update incorrect information manipulating the PostgreSQL spatial table and see the results of its operations by GeoServer maps at once.

3.4.9 Installing Ubuntu inside Windows using VirtualBox

Introduction

VirtualBox allows you to run an entire operating system inside another operating system. The minimum RAM should be 512 MB but 1 GB of RAM or more is recommended.

Comparison to Dual-Boot

A dual-boot allows one, at boot time, to decide which operating system to use. Installing Ubuntu on a virtual machine inside of Windows has a lot advantages over a dual-boot (but also a few disadvantages).

Advantages of virtual installation

- The size of the installation doesn't have to be predetermined. It can be a dynamically resized virtual hard drive.
- You do not need to reboot in order to switch between Ubuntu and Windows.
- The virtual machine will use your Windows internet connection, so you don't have to worry about Ubuntu not detecting your wireless card, if you have one.
- The virtual machine will set up its own video configuration, so you don't have to worry about installing proprietary graphics drivers to get a reasonable screen resolution.
- You always have Windows to fall back on in case there are any problems. All you have to do is press the right Control key instead of rebooting your entire computer.
- For troubleshooting purposes, you can easily take screenshots of any part of Ubuntu (including the boot menu or the login screen).
- It's low commitment. If you later decide you don't like Ubuntu, all you have to do is delete the virtual hard drive and uninstall VirtualBox.

Disadvantages of virtual installation

- In order to get any kind of decent performance, you need at least 512 MB of RAM, because you are running an entire operating system (Ubuntu) inside another entire operating system (Windows). The more memory, the better. It is recommended to have at least 1 GB of RAM.

- Even though the low commitment factor can seem like an advantage at first, if you later decide you want to switch to Ubuntu and ditch Windows completely, you cannot simply delete your Windows partition. You would have to find some way to migrate out your settings from the virtual machine and then install Ubuntu over Windows outside the virtual machine.
- Every time you want to use Ubuntu, you have to wait for two boot times (the time it takes to boot Windows, and then the time it takes to boot Ubuntu within Windows).

Installation Process

The VirtualBox was obtained from the internet and installed in the same way as any normal Windows program. The Ubuntu disk image (**.iso** file) was obtained and installed according to the screen shots given below:

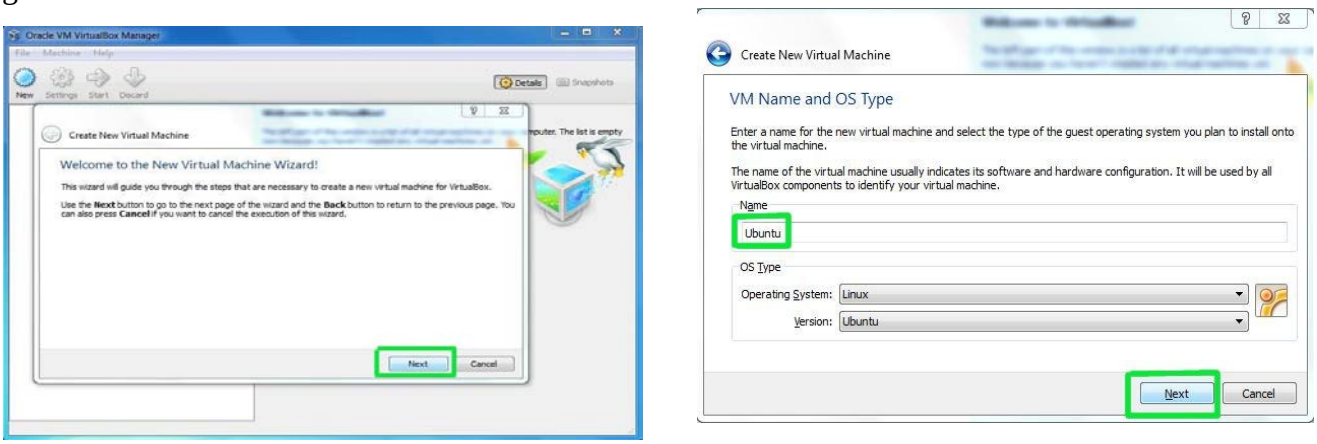

Figure 7: Ubuntu Disk Image

After VirtualBox launch from the Windows Start menu, click on **New** to create a new virtual machine. When the New Virtual Machine Wizard appears, click **next**. The machine name can be anything but since we are installing Ubuntu, it makes sense to call it **Ubuntu**. It should also be specified that the operating system is **Linux**.

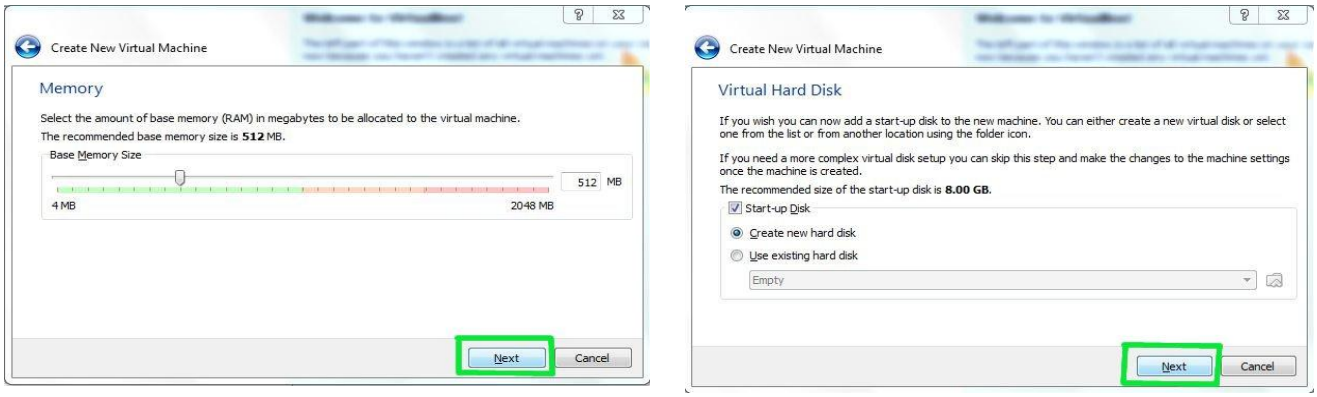

Figure 8: Virtual Machine Creation

VirtualBox will try to guess how much memory (or RAM) to allocate for the virtual machine. If you have 1 GB or less of RAM, it would be wise to stick with the recommendation. If, however, you have over 1 GB, about a quarter the RAM or less should be fine. For example, if you have 2 GB of RAM, 512 MB is fine to allocate. If you have 4 GB of RAM, 1 GB is fine to allocate. Click **Next**. There is need to *Create new hard disk* and then **Next** is clicked.

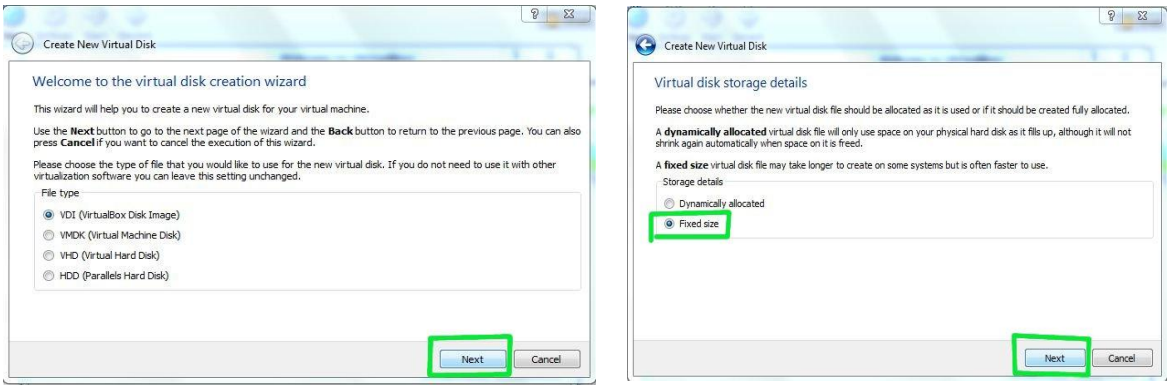

Figure 9: New Hard disk Creation

Click **next** again. Theoretically, a dynamically expanding virtual hard drive is best, because it'll take up only what is actually use. Sometimes there are situations, though, when installing new software in a virtualized Ubuntu, in which the virtual hard drive just fills up instead of expanding. So it is recommended picking **Fixed-size storage**.

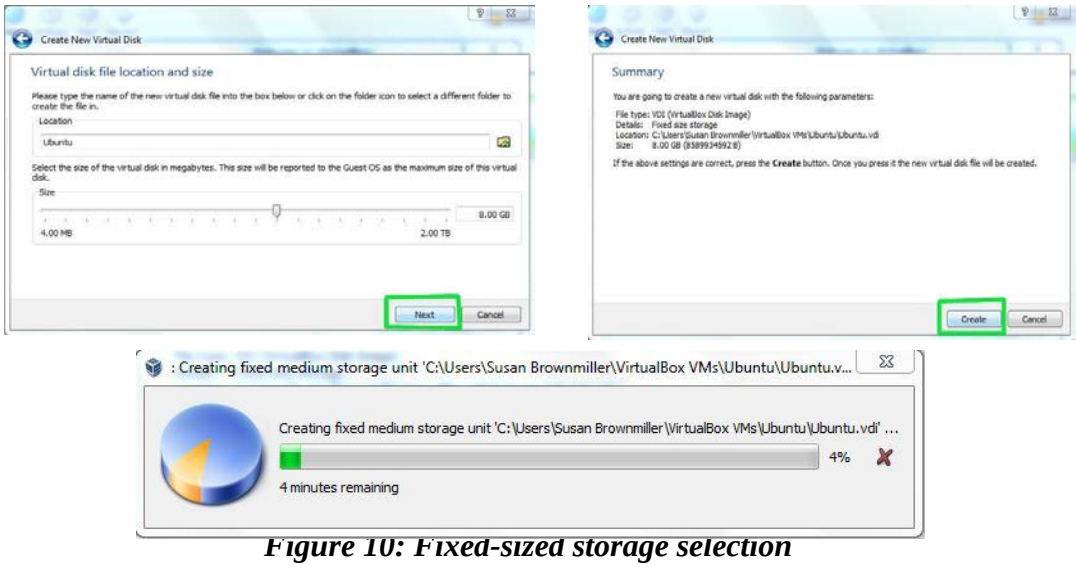

Figure 10: Fixed-sized storage selection

Ubuntu's default installation is less than 3 GB. Since we plan on adding software or downloading large files in the virtualized Ubuntu, some buffer is necessary. Click **Create** and wait for the virtual hard drive to be created. This is actually just a very large file that lives inside of your Windows installation.

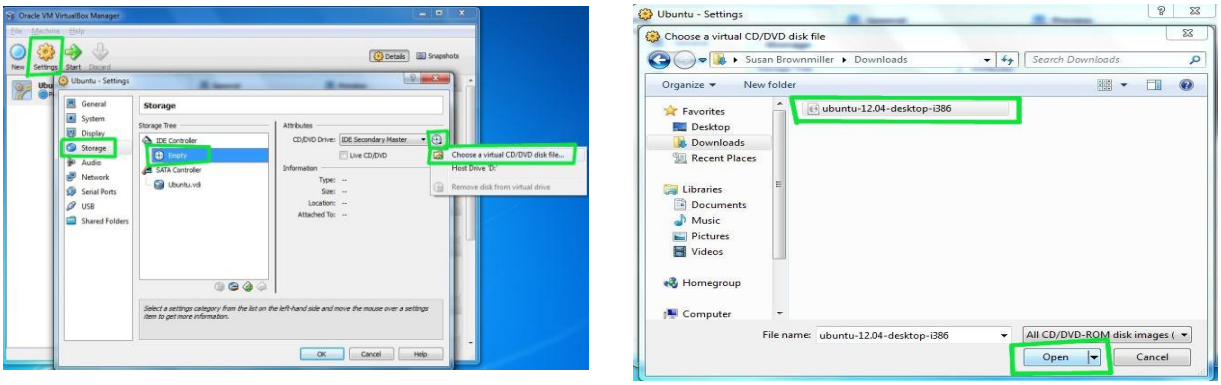

Figure 11: Fixed-sized storage buffering

The next thing to do to make the (currently blank) virtual hard drive useful is to add the downloaded Ubuntu disk image (the .iso) boot on your virtual machine. Click on **Settings** and **Storage**. Then, under *CD/DVD Device*, next to *Empty*, you'll see a little folder icon. Click that. Select the Ubuntu that you downloaded earlier (**.iso**).

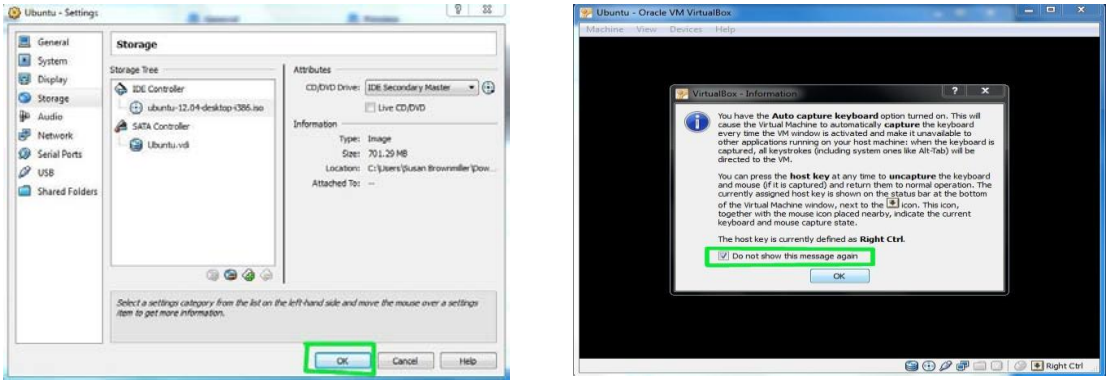

Figure 12: Making virtual hard bootable

Once selected, **OK** is clicked. Then the virtual machine is double-clicked to start up. There are a bunch of random warnings/instructions about how to operate the guest operating system within VirtualBox. These are marked not to be seen again.

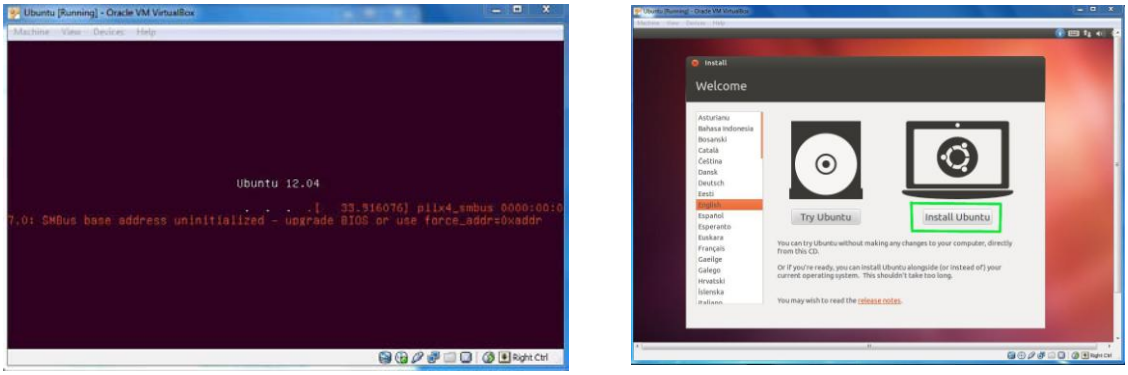

Figure 13: Operating guest operating System within VirtualBox

Wait for Ubuntu to boot up. Once it's started up, the instructions are straight forward as per the screenshot given here below:

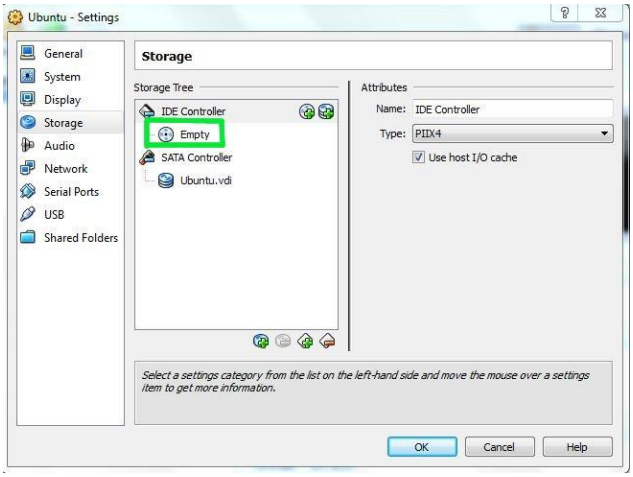

Figure 14: Using the virtualized installation

Afterwards, in order to use the virtualized installation there is need to double-check that the *CD/DVD Device* entry is **Empty** again.

3.4.10 Installing & Configuring PostgreSQL under Ubuntu Linux

PostgreSQL, often simply Postgres, is an object-relational database management system (ORDBMS) available for many platforms including Linux, FreeBSD, Solaris, MS Windows and Mac OS X. It is released under the PostgreSQL License, which is an MIT-style license, and is thus free and open source software.

PostgreSQL Installation:

Open the terminal and type following command to install PostgreSQL

sudo apt-get install postgresql

PostgreSQL Configuration:

After successful installation of postgresql, you can configure postgresql using configuration file /etc/postgresql/9.1/main/postgresql.conf (9.1 is the version of postgresql installed under Ubuntu). By default, connection via TCP/IP is disabled due to which users will not be able to access PostgreSQL server from another computers. To enable TCP/IP Connection the file /etc/postgresql/9.1/main/postgresql.conf is edited to make the following changes:

Change #listen_addresses = localhost to....

listen_addresses =192.168.1.1

and #password_encryption = on to...

password_encryption = on

Setting up PostgreSQL Users and Password:

NOTE: All the commands below are executed as the postgres privileged user.

Creating the user

At the terminal the command *createuser* and a few questions are answered to create PostgreSQL user

sudo -u postgres createuser

Enter name of role to add: linuxpoison

Shall the new role be a superuser? (y/n) n

Shall the new role be allowed to create databases? (y/n) n

Shall the new role be allowed to create more new roles? (y/n) n

Creating User Role

To create the PostgreSQL Database, the command *createdb* was used while the command to create the database was: *sudo -u postgres createdb linuxdb*

Creating Database

This is to grant access to the user of the database:

And last, using the *psql* command, the password is set for the user and accesses is granted:

sudo -u postgres psql

postgres=# alter user linuxpoison with encrypted password 'password';

Altering Role

postgres=# grant all privileges on database linuxdb to linuxpoison;

Grant

Now, on the client Linux (Ubuntu) machine, type following command to install PostgreSQL client:

sudo apt-get install postgresql-client

After successful installation of the postgresql client on the client machine, we connect to the server with the following command:

psql -h <postgresql_server_name> <database_name> <username>

After inserting the password, PostgreSQL is accessed with line commands. The following command (from terminal) to control the PostgreSQL server is used:

Start the service: /etc/init.d/postgresql start

Stop the service: /etc/init.d/postgresql stop

Know the status: /etc/init.d/postgresql status

Restart the service: /etc/init.d/postgresql restart

3.4.11 Installing GeoServer on Ubuntu 10.04

Following the steps below, the binary release of GeoServer running on 10.04 Ubuntu is installed. This should also work generically for other versions of Ubuntu.

1) **Installing necessary supporting libraries and applications using the terminal**

   sudo apt-get update

   sudo apt-get install gdal-bin openjdk-6-jdk openjdk-6-jre python-gdal unzip

2) **Downloading the latest stable or latest release of GeoServer.**

   cd ~wget http://downloads.sourceforge.net/geoserver/geoserver-2.1-RC4-bin.zip

3) **Extracting the release into the directory of choice.**

In my case I chose '/opt' since a different directory and release is chosen, it must be substituted accordingly.

cd /opt

sudo unzip ~/geoserver-2.1-RC4-bin.zip

3.4.12 Setting up a system for running geoserver.

1) **Creating a symlink**

We want *'/opt/geoserver'* to point to *'/opt/geoserver-RELEASE'* so that we can easily upgrade GeoServer at a later date.

sudo ln -s /opt/geoserver-2.1-RC4 /opt/geoserver

2) **Downloading GeoServer extensions.**

Follow the extension download link for whichever version you downloaded in the previous step here. Grab any extensions you want to install. I grabbed the following;

1. MySQL Data Store
2. GDAL Coverage Store
3. OGR output format

After downloading the extensions they are extracted to '/opt/geoserver/webapps/geoserver/WEB-INF/lib'.

cd ~

mkdir geoserver_extensions

cd geoserver_extensions

wget http://downloads.sourceforge.net/geoserver/geoserver-2.1-RC4-mysql-plugi...

wget http://downloads.sourceforge.net/geoserver/geoserver-2.1-RC4-gdal-plugin...

wget http://downloads.sourceforge.net/geoserver/geoserver-2.1-RC4-ogr-plugin.zip

*find . -name *.zip -exec unzip -o {} \;*

*sudo cp -rp *.jar /opt/geoserver/webapps/geoserver/WEB-INF/lib/*

3) Adding a GeoServer user and group

Next we create a group and user that GeoServer will run as:

sudo addgroup --system geoserver

sudo adduser --system --ingroup geoserver --no-create-home --disabled-password geoserver

4) Setting up startup script

In order to start geoserver automatically at startup we need an *init* script. I used the Debian/Ubuntu script located here.

cd /opt/geoserver/bin

sudo wget -O initd.sh http://docs.geoserver.org/latest/en/user/_downloads/geoserver_deb

sudo ln -s /opt/geoserver/bin/initd.sh /etc/init.d/geoserver

sudo chmod +x ./initd.sh

This script needs one slight modification to be better suited to Ubuntu. I changed the following line

 # Default-Stop: S 0 1 6

to

 # Default-Stop: 0 1 6

This is line 7 of the file.

5) Changing ownership of the GeoServer directory

The GeoServer install directory should be owned by the GeoServer user and group we just created.

sudo chown -R geoserver:geoserver /opt/geoserver-2.1-RC4/

6) Setting the default startup parameters

I then created a new file *'/etc/default/geoserver'*. The commented lines, starting with '#', indicate the default as provided in the startup script.

```
#USER=geoserver
#GEOSERVER_DATA_DIR=/home/$USER/data_dir
GEOSERVER_DATA_DIR=/opt/geoserver/data_dir
#GEOSERVER_HOME=/home/$USER/geoserver
GEOSERVER_HOME=/opt/geoserver
#PATH=/usr/sbin:/usr/bin:/sbin:/bin
#DESC="GeoServer daemon"
#NAME=geoserver
#JAVA_HOME=/usr/lib/jvm/java-6-sun
#DAEMON="$JAVA_HOME/bin/java"
JAVA_HOME=/usr/lib/jvm/java-1.6.0-openjdk
DAEMON="$JAVA_HOME/bin/java"
```

#JAVA_OPTS="-Xms128m -Xmx512m"

JAVA_OPTS="-Xms128m -Xmx512m -server"

#PIDFILE=/var/run/$NAME.pid

#SCRIPTNAME=/etc/init.d/$NAME

7) Setting the geoserver to launch on startup

sudo update-rc.d geoserver defaults

8) Setup log directories

sudo mkdir /opt/geoserver/webapps/geoserver/data/logs

sudo chown geoserver:geoserver /opt/geoserver/webapps/geoserver/data/logs/

9) Configuring OGR based WFS output format

This allows geoserver to export a bunch of formats that geoserver does not handle natively. The conversion is done through the OGR library. The following XML lets geoserver know where to find the ogr2ogr binary and the GDAL_DATA directory. Add the following XML to *'/opt/geoserver/data_dir/ogr2ogr.xml'*

```
<OgrConfiguration>
 <ogr2ogrLocation>/usr/bin/ogr2ogr</ogr2ogrLocation>
 <gdalData>/usr/share/gdal16</gdalData>
 <formats>
  <Format>
   <ogrFormat>MapInfo File</ogrFormat>
   <formatName>OGR-TAB</formatName>
   <fileExtension>.tab</fileExtension>
  </Format>
  <Format>
   <ogrFormat>MapInfo File</ogrFormat>
   <formatName>OGR-MIF</formatName>
   <fileExtension>.mif</fileExtension>
   <option>-dsco</option>
   <option>FORMAT=MIF</option>
  </Format>
  <Format>
   <ogrFormat>CSV</ogrFormat>
   <formatName>OGR-CSV</formatName>
   <fileExtension>.csv</fileExtension>
   <singleFile>true</singleFile>
   <mimeType>text/csv</mimeType>
  </Format>
```

```
  <Format>
    <ogrFormat>KML</ogrFormat>
    <formatName>OGR-KML</formatName>
    <fileExtension>.kml</fileExtension>
    <singleFile>true</singleFile>
    <mimeType>application/vnd.google-earth.kml</mimeType>
  </Format>
 </formats>
</OgrConfiguration>
```

10) Starting the geoserver

sudo /etc/init.d/geoserver start

11) Testing the geoserver

Now if everything has gone according to plan the geoserver should be running and accessible on port 8080. The default username and password are *gis* and *gis* respectively.

12) Trouble-shooting

Note that it can take geoserver quite a bit of time to actually bind to port 8080 and start responding. If you attempt to access geoserver on port 8080 and get an error message then wait a few minutes and try again. You can check to see if the geoserver is running by doing the following

ps aux | grep java

This should show something like the following indicating that the geoserver process is running.

5487? Sl0:21 /usr/lib/jvm/java-1.6.0-openjdk/bin/java -Xms128m -Xmx512m -server -DGEOSERVER_DATA_DIR=/opt/geoserver/data_dir -Djava.awt.headless=true -jar start.jar

If geoserver process is running one can check to see if it has opened port 8080 and is listening for requests with the following:

sudo lsof | grep TCP | grep geoserver

The result should be as follows:

```
java 5487 geoserver  237u    IPv6    20420    0t0    TCP *:46378 (LISTEN)
java 5487 geoserver  249u    IPv6    20435    0t0    TCP *:http-alt (LISTEN)
```

If you do not see the above but the java process is running just wait a few minutes as it can sometimes take a while for the geoserver to bind to port 8080.

3.4.13 About WEB GIS

WEB-GIS is an open source platform that facilitates the creation, sharing, and collaborative use of geospatial data. This web based application aims to surpass existing spatial data infrastructure solutions by integrating robust social and cartographic tools.

At its core, WEB-GIS is built on a stack based on GeoServer, PostGIS/Postgres, Django, and GeoExt that provides a platform for sophisticated web browser spatial visualization and analysis. It achieves this through a simple to use map composer and viewer that facilitates analysis, and reporting.

3.4.14 Geospatial data storage

The created PNVi WEB-GIS allows the user to upload vector and raster data in their original projections using a web form. Vector data is uploaded in ESRI Shapefile format and satellite imagery and other kinds of raster data are uploaded as GeoTIFF.

After the upload is finished, the user is presented with a form to fill in the metadata that describes what the data uploaded is all about.

Similarly, WEB GIS provides a web based style, that lets the user change how the data looks and preview the changes in real time.

3.4.15 Data Upload and Map creation

Once the data has been uploaded, WEB GIS lets the user search for it geographically or via keywords and create maps.

All the layers are automatically re-projected to web Mercator for maps display, making it possible to use different popular base layers, like Google Map Layers or Bing Map layers.

Once maps are saved, it is possible to embed them in any webpage or get a PDF version for printing.

3.4.16 Architecture Overview

This section provides insight about the components of WEB GIS and the way they have been used to create the final product. This information is mostly important to the WEB GIS Administrator or a GIS developer who would want to understand the architecture of WEB GIS.

The main components used are:

1. **The Geospatial Data Manager: GeoServer**

GeoServer provides an *OGC* compatible data store that can speak to *WMS*, *WFS*, *WCS* and others in common formats like *GML*, *GeoJSON*, *KML* and *GeoTIFF*. It can be connected to different spatial backends including *PostGIS*, *Oracle Spatial*, ArcSDE and others.

2. **The Catalog: GeoNetwork**

GeoNetwork provides a standard catalog and search interface based on *OGC* standards. It is used via the *CSW* interface to create and update records when they are accessed in GeoNode.

3. **The Website: ASL GIS Django Site**

This is a Django based project that allows the user to easily tweak the content and look and feel and to extend WEB-GIS to build Geospatial. It includes tools to handle user registration and accounts, avatars, and helper libraries to interact with GeoServer and GeoNetwork in a programmatic and integrated way. There is a wide range of third party apps that can be plugged into a WEB GIS based site including tools to connect to different social networks, to build content management systems and more.

4. **The Map Composer: WEB GIS Client**

The main map interface for GeoNode is the Map Composer / Editor. It is built on top of *GeoExt* and uses OpenLayers, GXP

It talks to the other components via *HTTP* and *JSON* as well as standard *OGC* services.

WEB GIS provides a number of facilities for interactivity in the web browser built on top of several high-quality JavaScript frameworks:

- ExtJS for component-based UI construction and data access
- OpenLayers for interactive mapping and other geospatial operations
- GeoExt for integrating ExtJS with OpenLayers
- GXP for providing some higher-level application building facilities on top of GeoExt, as well as improving integration with GeoServer.
- And a WEB GIS -specific framework to handle some pages and services that are unique to WEB GIS.

5. **PostgreSQL & PostGIS/ArcSDE: Database**

The database component manages data and configuration information for GeoNode/Django, GeoNetwork and GeoServer. All of these tables and data are stored within the GeoNode database on PostgreSQL. GeoServer can use either PostGIS or ArcSDE to store and manage spatial vector data. (Each layer is stored as a separate table.) When PostGIS is used as the spatial database for GeoServer, the layers are also stored within the WEB GIS database.

6. Dependencies

Below is a list of the applications GeoNode relies on:

- PostgreSQL (Database Platform)
- PostGIS (Adds Spatial Functionality to the database)
- GeoTools (Contains GIS functionality e.g. symbology)
- GeoServer (Platform to render the Maps to the Web GIS)
- GeoWebCache (Cache to store files for quick retrieval)
- Mapfish printing module (Printing Functionality)
- OpenLayers (Add mapping capability e.g. panning, zooming, searching etc)
- GeoExt (For designing the interface)
- Django (Scripting language)

7. JavaScript in WEB GIS

The following concepts are particularly important for developing on top of the WEB GIS JavaScript framework.

- **Components** - Ext components handle most interactive functionality in "regular" web pages. For example, the scrollable/sortable/filterable table on the default Search page is a Grid component. While WEB GIS does use some custom components, familiarity with the idea of Components used by ExtJS is applicable in WEB GIS development.

- **Viewers** - Viewers display interactive maps in web pages, optionally decorated with Ext controls for toolbars, layer selection, etc. Viewers in WEB GIS use the GeoExplorer base class, which builds on top of GXP's Viewer to provide some common functionality such as respecting site-wide settings for background layers. Viewers can be used as components embedded in pages, or they can be full-page JavaScript applications.

- **Controls** - Controls are tools for use in OpenLayers maps (such as a freehand control for drawing new geometries onto a map, or an identify control for getting information about individual features on a map.) GeoExt provides tools for using these controls as ExtJS "Actions" - operations that can be invoked as buttons or menu options or associated with other events.

3.5 APPLICATION DEVELOPMENT

The initial Requirement Analysis contained some applications of a complex nature. However, most of initial applications functionalities were straightforward and were implemented using the basic functionality that is part of the Web-GIS software (e.g., display). The more complex applications that are not supported by the basic functions of Web GIS were coded or programmed. Ease of use, user-friendliness, and reducing the volume of data transfer were the critical issues to be considered in the development. The user interface was programmed using HTML, Java Script, and Open Layer. At this point in the Web-GIS development process, the Web-GIS hardware and open source software were acquired (downloaded) and data conversion completed.

3.5.1 System development

PNVi Web-GIS Viewer was developed on the available Open Source Web-GIS framework of Python, HTML, Javascript and PHP. These tools use a single and scalable architecture. They also provide the basis for developers to write programs in .NET (VB and C#) and JAVA platform for customized applications.

The geodatabase (under PostGIS) data model replaces the georelational data model that has been used for coverages and shapefiles, two older data formats from ESRI, Inc. These two data models differ in how geographic and attribute data are stored. The georelational data model stores geographic and attribute data separately in a split system: geographic data ("geo") in graphic files and attribute data ("relational") in a relational database.

A geodatabase uses tables to store geographic data as well as non-geographic data. A table consists of rows and columns. Each row corresponds to a feature, and each column or field represents an attribute. A table that contains geographic data has a geometry field, which distinguishes the table from tables that contain only non-geographic data.

The database engine used is PostgreSQL server 9.1 and PostGIS (spatial database engine).

The development environment is Python integrated with PHP and GeoNode (GIS framework).

3.5.2 Unit Testing

Unit testing was carried out during the development time against a prototype database. Two methods black box and white box testing were employed.

Black Box Testing

This is a method of testing software that test the functionality of an application as opposed to its internal structures. In this type of testing for PNVI GIS Viewer Decision tables were used to bring out a precise yet compact way to model complicated logic. Decision tables, like flowcharts and if-then-

else and switch-case statements, associate conditions with actions to perform were analyzed to ensure that the functionalities come out clearly.

White Box Testing

This is a method of testing software that tests internal structures or workings of an application. Control workflow was employed which refers to the order in which the individual statements, instructions, or function calls of a declarative PNVI GIS Viewer Program were evaluated.

3.5.3 Integration and System Testing

Individual PNVI GIS Viewer software modules were combined and tested as a group. The multiple integrated modules passed unit testing and were used as input and tested their required interactions. Testing was done against the main PNVI Database as opposed to unit testing which was carried out on a prototype database on the development machine. Test will continue by the PNVI staff for at least 6 months to ascertain that the system functions well.

3.5.4 Deployment and Maintenance

The application is deployed in an online framework using GeoNode technology. Hosting is at the WWF Eastern and Southern Africa Regional Programme Offices server. Maintenance is annual by me and the WWF's IT department. Any bugs will be fixed and data updates will be a continuous process.

3.5.5 Requirements and prerequisites

Your computer must meet these minimum system requirements to install the PNVi Web-GIS system:

1. **Processor** Pentium® 1 GHz minimum
2. **Operating system**
    - Microsoft Windows 2000 Professional with Service Pack 3 or higher
    - Microsoft Windows Server 2003
    - Microsoft XP Home Edition with Service Pack 1 or higher
    - Microsoft XP Professional Edition with Service Pack 1 or higher
    - Microsoft Vista

    (Hereafter collectively referred to as Microsoft Windows)
    - Ubuntu Linux
3. **RAM** 1 GB minimum

3.5.6 Setting up the Web-GIS Viewer users

The system administrator will create rights for users just like in any other SQL server database. Users should have the minimum requirements of **dbowner** and **dbreader**. Editors (DB Administrator and GIS Specialist) should have rights of **dbowner**, **dbreader** and **dbwriter**.

3.6 DATA SOURCES, PROCESSING AND LOADING

The main objects of interest of Virunga National Park's Web-GIS are the demarcation of the park boundary to avoid conflicts, biodiversity studies, socio-economic surveys, vegetation mapping and their cataloguing and cartographic displaying, so the entire system was designed and implemented for receiving and using as well as sharing these data. There is also a second group of data, which gives information about the whole Virunga National Park and allows the analysis of the conservation features in the territory where they stand. Later on, we indicate wildlife census data and environmental and territorial data such as Park boundaries, wildlife observation works, vegetation and habitat diversity and infrastructure.

3.6.1 Sources and processing

At the beginning of the work there wasn't any digital data describing Virunga National Park (PNVi) site and remains, so a great deal of data acquisition and data processing work was necessary. In particular my works over the years in Eastern DRC have ensured that I gathered these datasets and have been slowly scanning, georeferencing and digitizing them. The data is in the following categories:

-Digital data and *in-situ* inspections;

-Other cartographic and textual/attribute information;

-GPS surveys.

-Satellite remotely-sensed imagery (Landsat, ASTER and SPOT)

-Numerous old (1958/1959) and new (2004-2009) aerial photographs

There existed also enormous amounts of data that was available in Universities in Belgium, the former colonial power in the Democratic Republic of the Congo. Universities of Louvain (UCL) and Ghent turned out to be very resourceful especially for historical datasets that gave a picture of how Virunga National Park has been changing over time.

3.6.2 Other cartographic and textual information:

For the completion, updating and acquisition of information about conservation works, it was also necessary to use other sources in addition to PNVi catalogues and maps. One of these is the University of Louvain (UCL's) scanned map sheets at scale 1:50,000, which will be useful for historical changes and boundary descriptions of the PNVi region of interest. Other used cartographies are ancient Topo-cadastral maps of *Congo-Belge* (the colonial era of the Congo, 1935). Lastly archaeological and environmental books and papers were examined to give the territorial knowledge

and to aid in database conceptual design steps. In particular, I concentrated on information about biodiversity, livelihoods and trends in Virunga National Park territory, population and landuse.

3.6.3 GPS survey:

A Global Positioning System (GPS) is used to gather location information of features on the ground. GPS receivers provide quality checks on how georeferenced data in the GIS Database (DB) overlays with GPS-gathered data with the location of features in the DB such as roads, buildings, or habitat edge. To do this, one needs to collect data with GPS, import it into Quantum GIS, and overlay it with data from the DB.

A GPS campaign is usually divided into the following sections:

1. GPS Planning: To decide what we want to collect so as to come up with a survey design. In addition, we prepare the form for field data collection.
2. GPS Configuration: To configure the GPS receiver so it records coordinates in the correct coordinate system and units.
3. Data Collection: switch on our GPS gadgets. We proceed to collect various points as per our design
4. GIS Integration: Then download the points to our laptops. We can do this in one of two ways: If we have a Garmin GPS we can use the extension DNR Garmin. If we don't, we will import the data as a table.

In this study, some data was collected and fed into the database after being downloaded and pre-processed using Q-GIS.

4.0 RESULTS, SYSTEM COMPONENTS AND FUNCTIONALITIES

4.1 THE DATASETS MOBILIZED

The general data flow for PNVi WEB-GIS Viewer is represented by the basic structure and setup of the WEB-GIS for WWF and Partners.

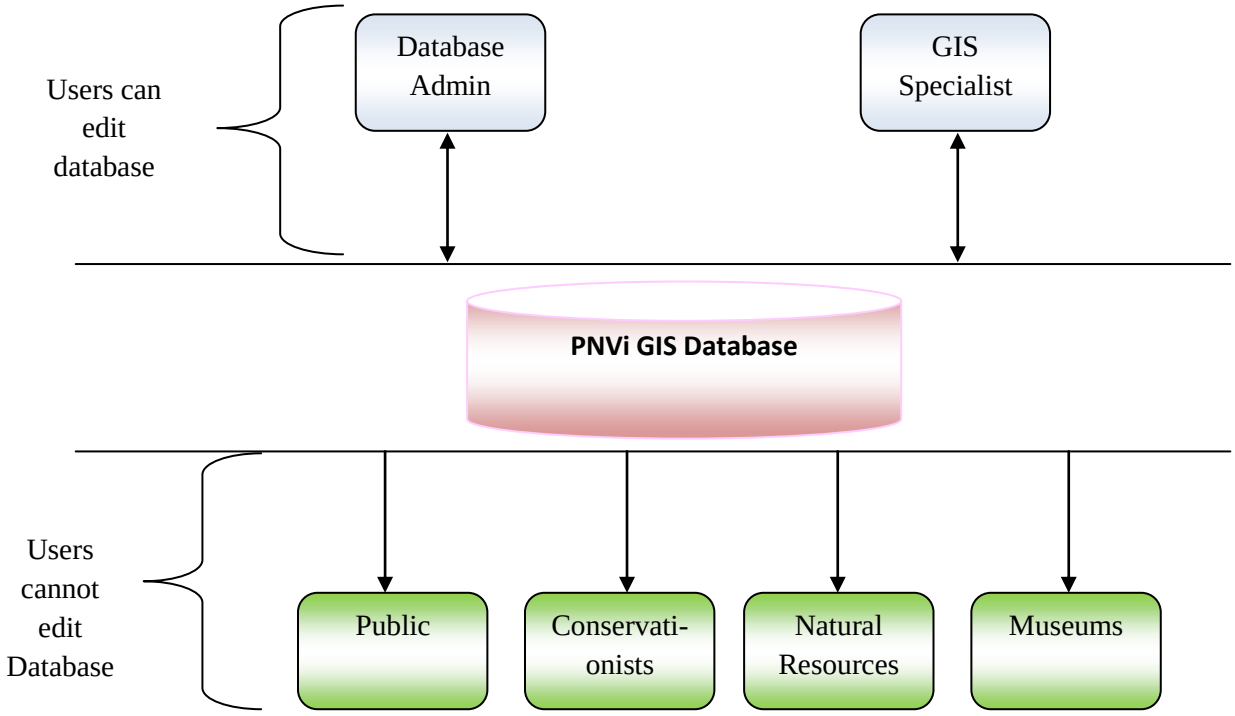

Figure 15: The General Data Flow of the System

The figure above (Figure 15) shows the basic structure and setup of the Virunga National Park Web-GIS Database. The Database Administrator and the GIS specialist have the exclusive rights to use and edit the database. They connect to the database with administrative rights and roles through the custom PNVi Web-GIS map viewer, while the others have use rights only. They connect to the database through the Web-Browser and can view and, depending on their roles, add data but not edit.

The following datasets were mobilized, pre-processed and uploaded to the PNVi online Database:

1. SPOT 5 and Landsat 7 satellite-remotely sensed imagery for the years 2004 to 2007 for SPOT and 1972 to 2002 for the Landsat imagery. The SPOT satellite imagery were of the following spectral characteristics:
    a. Spatial resolution = 5m
    b. Spectral resolution = 4 bands, RGB and NIR
    c. Temporal resolution = annual
    d. High quality radiometric resolution

The Landsat 7 imagers were of the following characteristics:

a. Spatial resolution = 28.5m
b. Spectral resolution = 7 bands
c. Temporal resolution = 10 years
d. Of high radiometric resolution

2. Topographic map sheets at scale of 1:50,000 drawn and compiled by the colonial Belgian authorities in 1935 and 1948. These were scanned using large format colour scanner, georeferenced and colour separated for the different features. These maps were used by the Belgian government to establish the legal limit of the Virunga National park and have been extremely useful in the interpretation of the legal text that describes the limits of the park. Without these topographic maps, it would have been nearly impossible to re-establish the vastly encroached park boundaries. In the PNVi database, these datasets form an integral part of the archives and for change analysis
3. Aerial photographs that were done by the Belgian authorities in 1958. These black and white aerial photos are instrumental in the detection of changes such as expansion of urban centres, expansion and sprouting up of the legal and illegal landing sites (fisheries), habitat changes as well as visual interpretation and re-establishment of the integrity of the park boundaries. With their discoverability on the platform that we have developed, they form an integral part of the system archives and historical documentation of what the situation used to be before the Rwandan genocide which led to the heavy encroachment of the park limits. They have also been useful in determining the buffer and hunting zones which had been established to prevent human-wildlife conflict and provide for sustainable extraction of natural resources by the communities living around the protected area.
4. Wildlife monitoring data. This dataset was developed by the rangers over a period of 5 years and involved documentation of wildlife sightings, be they life animals or carcasses of different types of species. Since the data has a spatial aspect, it is easy to monitor animal movements and therefore design wildlife corridors and advise populations to avoid human-wildlife conflicts which lead to destruction of crops and even human deaths. The dataset also helps in determination of species endemism and diversity and hence biodiversity sensitivity to economic development. When combined with animal census data, it helps to list endangered species and determine protection strategies.
5. Data on Hippos found in four major rivers feeding into that form part of the Lake Edward ecosystem i.e. Rivers *Rwindi, Ishasha, Rutshuru* and *Semuliki*. These datasets depict an

important loss of hippo population from a population of 30,000 individuals in 1974 to a mere 900 individuals in 2005.

6. Ranger posts. This dataset provides the spatial distribution of ranger posts and provides a picture of where to construct new posts to help in wildlife protection and surveillance.
7. Urban centres and Villages. The locations of civilizations are useful in habitat modelling as well as modelling of scenarios around natural resources exploitation. When ecosystem services are valued, distances to populations benefitting from these resources are key inputs to the model that help show benefits of the ecosystem services and tradeoffs to communities.
8. Signposts (*pancartes*). These datasets give the spatial locations of the signposts used to delimit the Virunga National Park boundaries.
9. Hydrology. Rivers, lakes and other waterbodies have been uploaded to the system for use in hydrological modelling for ecosystem services as well as impacts due to climate change on eFlows (environmental flows i.e. water quantities and qualities as well as discharge).
10. Infrastructure. Planning for livelihoods around protected areas will not be complete without datasets on infrastructure such as road, rail, port and airport networks as well as all other forms of transport and communications. Exploitation of natural resources is always a function of nearness of these resources to access points.
11. Data on community based natural resources management (CBNRM) intervention areas were also mobilized. These data show the distribution of the intervention sites where WWF is working with communities in a participatory approach to natural resource management.

Once these and other datasets were mobilized and pre-processed for integrity in terms of coordinate projection systems, map datums and map units as well as topological integrity for vector datasets, they were uploaded to the database and are being accessed, manipulated and used to make maps for decision-support as described in the following section that depicts results pertaining to the interface of the designed system as well as its functionalities which include map creation, querying, searching and uploading as well as downloading of shared data. The system is also designed to ensure that there is data quality control since those that upload and manipulate data have their credentials logged and should there be any queries or issue about data (which include copyright issues), the owner of the data can be contacted.

4.2 THE SYSTEM DESIGNED AND ITS FUNCTIONALITIES

4.1.1 Creating MAPS from the Web-GIS Viewer

Launch any web-browser and type in the URL (in our case www.wwfesarpogis.org). From the interface that appears, click CREATE MAP to make the map using any of the layers that have been uploaded to the database.

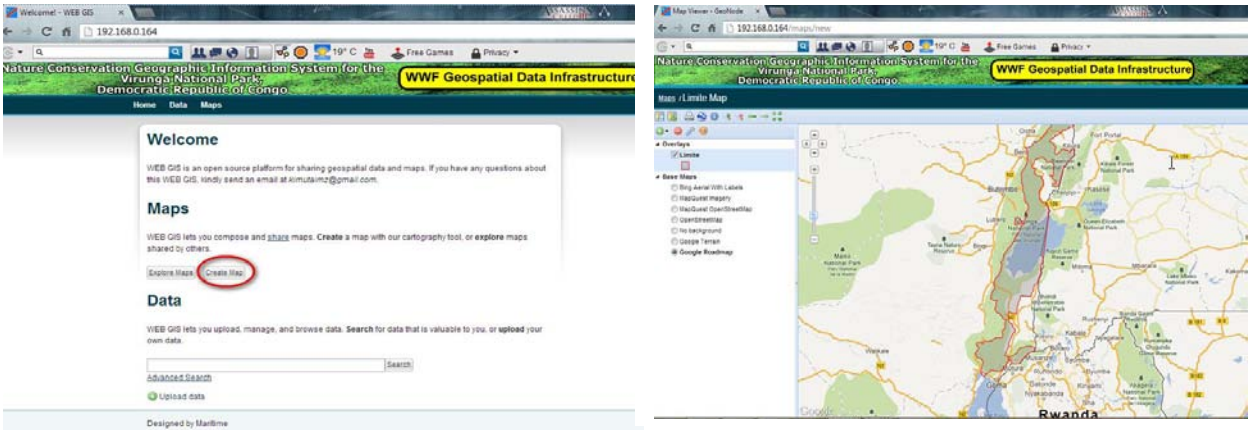

Figure 16: Map Creation Page

4.1.2 System Features

4.1.2.1 Home and Login Pages

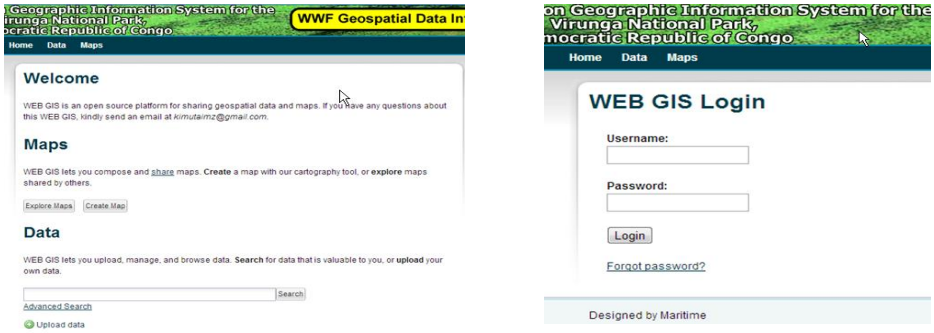

Figure 17: Home and Login Page

4.1.2.2 Data Upload and Metadata Creation Pages

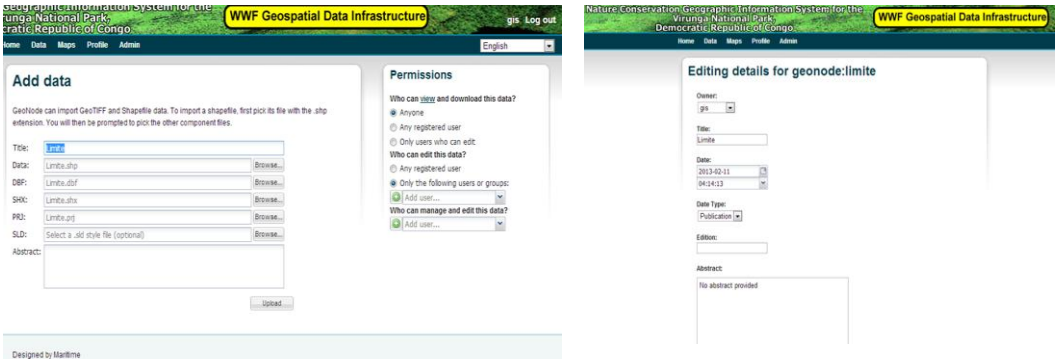

Figure 18: Data Upload and Metadata Creation Pages

4.1.2.3 Previewing Uploaded Data

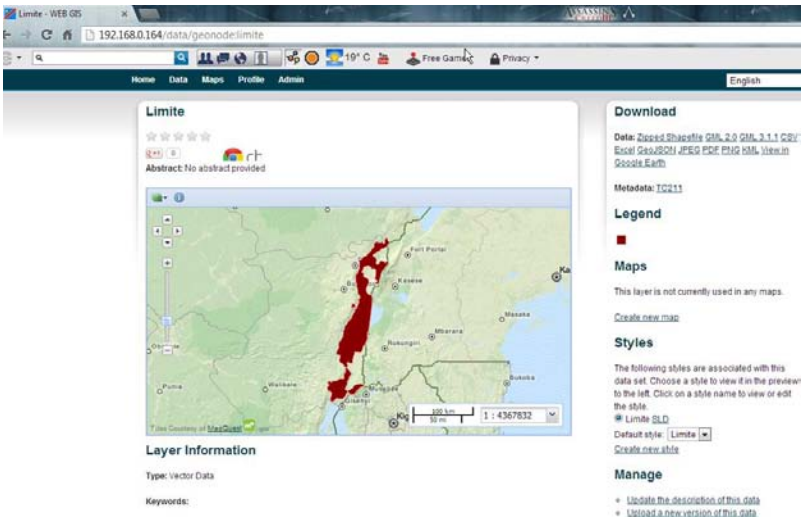

Figure 19: Previewing uploaded data

4.1.2.4 Data Downloading and Permissions Setting

Figure 20: Data downloads and usage permissions

4.1.2.5 Listing Uploaded Data

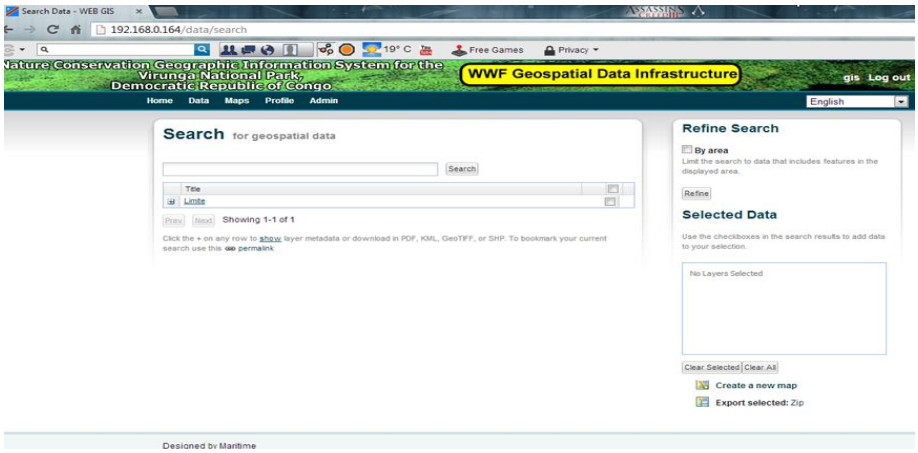

Figure 21: Listing uploaded data

4.1.2.6 Adding Layers when creating map

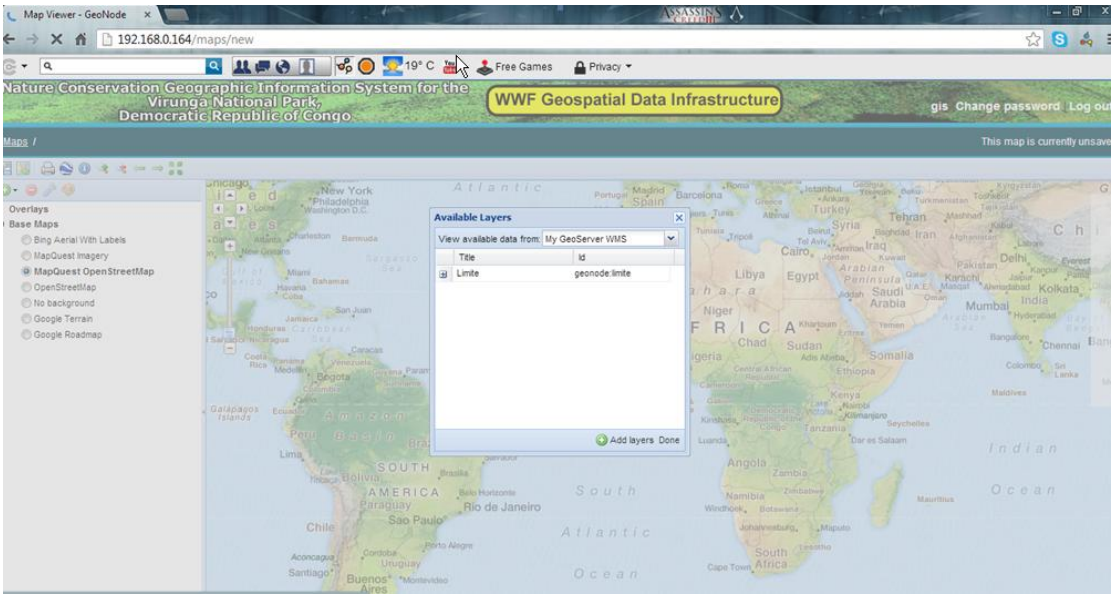

Figure 22: Adding layers when creating map

4.1.2.7 User profiles pages

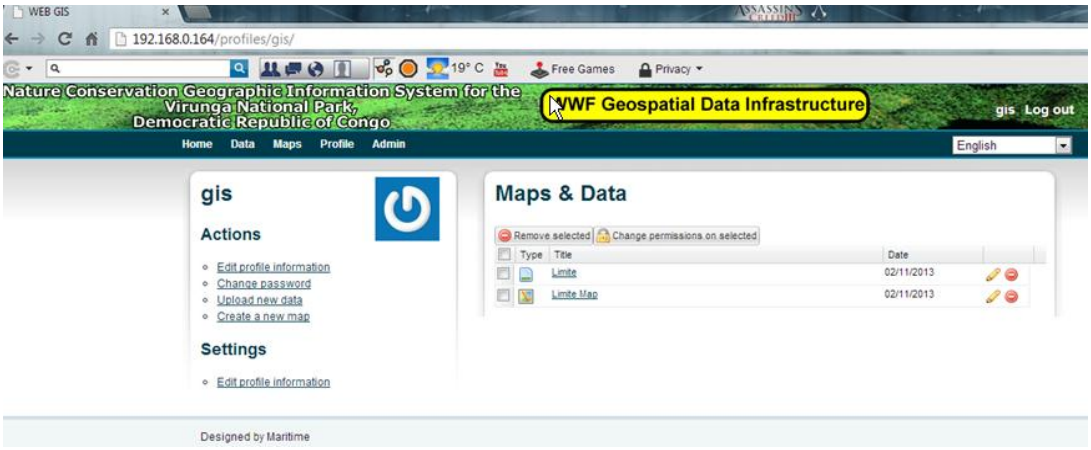

Figure 23: User profiles page

5.0 CONCLUSIONS & RECOMMENDATIONS

5.1 CONCLUSIONS

5.1.1 Achievement of the Objectives

The objectives that we set out to achieve in this project were achieved as follows:

5.1.1.1 Objective 1:

With regard to the first objective, which was concerned with data mobilization, evaluation and studying of the Virunga National Park strategic plans to obtain information on the data availability, data from various sources including the Université Catholique de Louvain and University of Ghent, both in Belgium, were obtained and cleaned for consistency and topological consistencies. Strategic and Management plans of the Virunga National Park were obtained from the Congolese Institute for Nature Conservation, the *Institut Congolais pour la Conservation de la Nature* (ICCN). These plans aided in understanding the various data types needed for conservation decision support which includes ways in which wildlife conservation interacts with livelihoods to create harmony.

A conclusion can be drawn from this achievement that for the success of the systems of this nature, there's need for concerted effort working with partners and museums in the western world who hold huge amounts of data and

5.1.1.2 Objective 2:

The second objective was concerned with the design and development as well as tuning and configuration of an enterprise geodatabase under PostGIS/PostgreSQL, the objective was implemented via Open Source platforms bearing in mind the limited money flowing into the world of nature conservation. The platform adopted was that employing GeoNode deployment and exploiting GeoServer Technology, PostgreSQL/PostGIS all these being Open Source Technologies.

5.1.1.3 Objective 3:

The third objective was concerned with data collection, collation and migration into the constructed database. Data that had been mobilized from many different sources, including universities in the home countries of former colonial masters (mostly Belgium), was migrated from flat file shapefiles and GeoTIFF into the well-constructed PostGIS/PostgreSQL database which is being accessed from a web browser as client.

5.1.1.4 Objective 4:

Eventually to address the final objective of the WebGIS application development, the application was developed, tested and deployed using GeoNode technology. Both Unit Testing and Integration and System Testing were done. Then the system was hosted and deployed online for the conservation

stakeholders to use. The URL was sent out to WWF and its partners to make use of the system and give feedback for continued improvement of its functionalities and usefulness. The URL is: www.wwfesarpogis.org.

5.1.2 Conclusions drawn

The Virunga National Park (PNVi) WEB-GIS can be considered a didactic and easily accessible tool for the study of conservation impacts on livelihoods and an example of GeoServer-PostGIS/PostgreSQL technology combination. However, PNVi WEB-GIS is always open to future improvements, both for data increases and system updating, upgrading and new functions and extensions implementation. These improvements are obviously necessary for a real and continuous use of PNVi Web-GIS, especially for being in step with conservation research, territorial and environment management. The following conclusions can be drawn from this study:

- One of the lessons learnt from this study was that it is always a time-consuming, cost and labor-intensive exercise to collect data. Most existing data on national parks in Africa, Virunga being one of the oldest, is found in European countries lying in hard copies or some of it digitized. To repatriate these datasets requires a lot of lobbying as well as connections to institutions in these former colonial masters' territories.
- Processing of collected data requires a lot of expertise in different types of data processing software that include image-processing software and vector data processing software such as ERDAS Imagine, ENVI, MicroStation, AutoCAD, IDRISSI, Definiens, PCI Geomatica, Q-GIS, etc. The expert needs to combine all these applications for different situations since while one may be good in image geometric and atmospheric corrections (for example), the other will be the best in image mosaicking.
- An online repository of the mobilized, digitized and calibrated (quality-controlled) datasets provides a means of storage of the data for posterity as well as enhancing discovery of these data for scientific research and knowledge enhancement. Once published online, the datasets can be used for value addition in development projects. Data owners can be cited and acknowledged through published materials such as scientific journals.
- Data made available over the internet increases access to them for use in decision-making such as in planning for socio-economic development project while taking into consideration biodiversity sensitivity and the fact that if we destroy nature, nature will destroy us. Nature conservation cannot happen without data that will inform deliberate steps to take in this effort.

5.2 RECOMMENDATIONS

From a commissioned WWF User Needs Assessment and discussion with WWF's management, it was recommended that there was need for an online data sharing and map making platform that can be used across WWF's countries of operation (Kenya, Uganda, Tanzania, Zambia, Zimbabwe, Mozambique, Rwanda and Eastern DRC). This case study of the Virunga National Park will go a long way in ensuring that this platform is constructed and deployed to aid WWF in all its efforts to conserve nature in a collaborative manner.

The improved functions of the application are map creation, map browsing, data searches, data queries, data uploads and downloads as well as data styling and conservation features thematic views, while saving a great deal on software costs since the system has been implemented in open source platforms. The system also ensures modernization of data sharing, avoiding old techniques of sending CDs and DVDs by courier as well as improved data backup and storage mechanisms.

Further recommendations include:

- Integration of mobile data collection and direct upload to the system. With many modern and robust tools for geospatial field data collection available in the market today, whether android-powered, Windows-CE powered, Apple-powered, etc it is going to be necessary to integrate mobile GIS in this web-based data sharing and visualization platform.
- There is need to implement the Quantum-GIS (QGIS) platform which is an open source application that talks to the PostGIS and retrieves data for more complex analyses and returns processed data for sharing through the infrastructure that we have built. This is strongly recommended for the system to become more and more complete and versatile. We however used the Q-GIS Open Source GIS application to pre-process the data for eventual upload to the database.
- For sustained data collection to feed into the system, a data collection strategy is recommended. One such strategy is to have a framework of regular GPS campaigns to monitor biodiversity and other indicators such as socio-economic factors in conservation and landcover change as well as environmental flows (eFlows). Figure 24 below shows a schematic for GPS data collection strategies that is recommended for updating the online database.

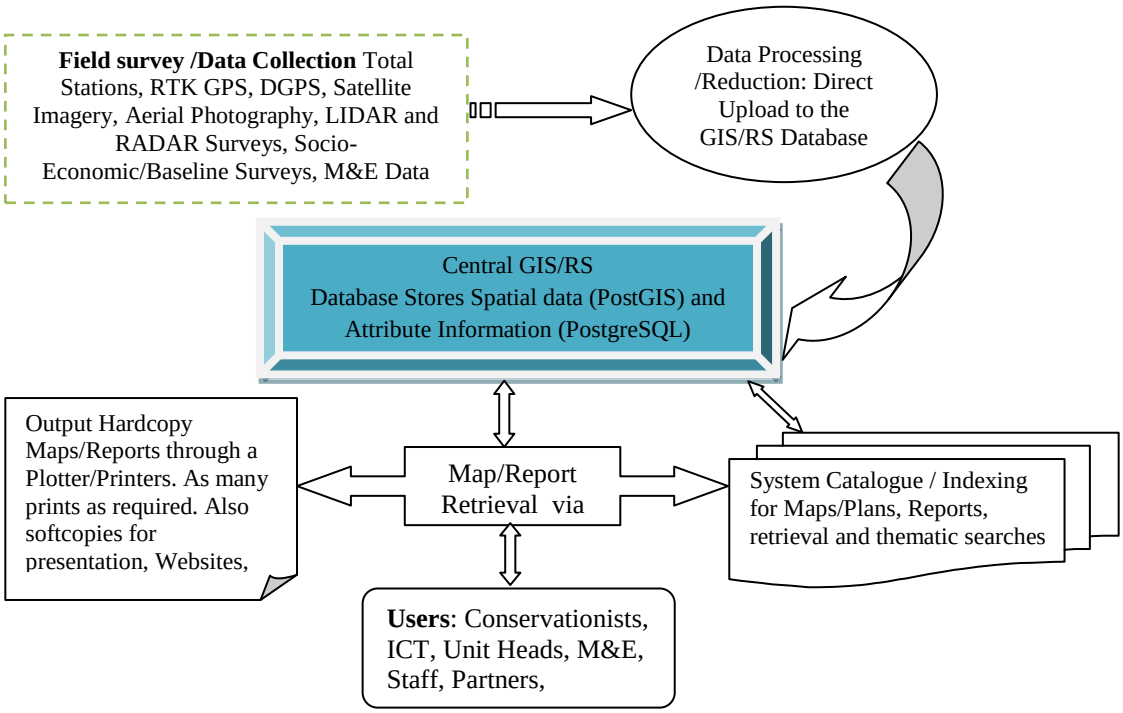

Figure 24: GPS survey data processing

- We recommend this Open Source GIS Technology to organizations that exploit spatial data for decision making and would like to reduce costs occasioned by the commercial software.
- This being a multi-disciplinary domain comprising experts in ecology, spatial systems and ICT, organizations need to plan well how the services of these different professionals will be engaged. While one expert may be competent in spatial systems design, the quality of ecological/conservation data will only be assured when an ecologist is implicated.

6.0 REFERENCES

Abiteboul, S., Hull, R. and Vianu, V. (1995) *Foundations of Databases*, Reading MA: Addison-Wesley.

Adam, N. R. and Gangopadhyay, A. (1997) *Database Issues in Geographic Information Systems. Vol. 6 of Advances in Database Systems pp. 413–422,* New York: Kluwer Academic Publishers.

Björn, Schilberg (2011) www.freegis.org.

Brian N. H. (2007) *Emerging Spatial Information Systems and Applications*, London: Idea Group Publishing.

Buchanan, H. (1994) *Geographic Information*: *The Source Book for GIS*: In D. R. Green, D. Rix, and Cadoux-Hudson, J. (1994) *Standards for Spatial Data Transfer, pp 254–261*. London: Taylor and Francis in cooperation with the Association for Geographic Information.

Cynthia, A.B. (2005) *Designing Better Maps: A Guide for GIS Users*, Redlands CA: ESRI Press

Erik, H. (2010), *Open Layers 2.10: Beginner's Guide*, Birmingham, Pack Publishing Ltd.

Erik, W. (2010) *Python Geospatial Development*, Birmingham: Packt Publishing Ltd.

Gary, S. (2012) *The Geospatial Desktop: Open Source GIS and Mapping*, Williams Lake BC, Canada: Locate Press.

Goodchild, M. and Jeansoulin, R. (1998) *Data Quality in Geographic Information: From Error to Uncertainty*. Paris: Hermes.

Languy, M. and deMerode, E. (2006) *Virunga; Survie du premier parc d'Afrique, pp13*, Tielt, Belgium: Lannoo.

Maguire, D., Goodchild, M. and Rhind, D. (1999) *Geographic Information Systems*. *2d ed., 2 vols*. London: Longman Scientific and Technical.

Neil, M. and Richard, S. (2005) *Beginning Databases with PostgreSQL, 2nd Edition: From Novice to Professional*, New York: Springer-Verlag.

Plewe, B. (1997) *GIS On-Line: Information Retrieval, Mapping, and the Internet*, Albany NY: On-Word Press, Delmar, Publishers.

Ramsey, P., Refractions Research Inc, (2003). *PostGIS Manual*. http://postgis.refractions.net/

Regina, O.O. and Leo, S.H. (2011) *PostGIS: In Action*, Stamford, CT: Manning Publications Co.

Samet, H. (1990) *The Design and Analysis of Spatial Data Structures*. Reading MA: Addison-Wesley.

Servigne, S. and Laurini, R. (1995) *Updating Geographic Databases Using Multi-Source Information*: *In Proc. ACM Intl. Symp. On Geographic Information Systems (ACM-GIS)*, Software Cartlab: www.sifet.it/cartlab.htm

Stonebraker, M. and Rowe, L. A. (1986) *The Design of Postgres: In Proc. ACM SIGMOD Intl. Symp. On the Management of Data, pages 340–355*.

The PostgreSQL Development Team (1998) *PostgreSQL Tutorial (release 6.3),* Indianapolis: Thomas Lockhart

Tyler, M. (2005) *Web Mapping Illustrated: Using Open Source GIS Toolkits*, Sebastopol, CA: O'Reilly Media Inc.

UNESCO (1999) *Biodiversity conservation in regions of armed conflict: Protecting World Heritage in the Democratic Republic of the Congo (DRC).* Project Review Form. UNESCO World Heritage Centre, Paris, France.

Whitten, Bentley, Dittman (2000) *Systems Analysis and Design Methods, 5th Edition*: Irwin/McGraw-Hill

www.maptools.org, [accessed on 24 August 2012]

7.0 APPENDIX

Source Code for Interface

```
<html>
<head>

   <title> Map Viewer - GeoNode </title>
   <link rel="shortcut icon" href="/static/theme/img/favicon.ico"/>

<link  rel="stylesheet"  type="text/css"  href="/static/geonode/externals/ext/resources/css/ext-all.css"
/>
<script type="text/javascript" src="/static/geonode/externals/ext/adapter/ext/ext-base.js"></script>
<script type="text/javascript" src="/static/geonode/externals/ext/ext-all.js"></script>

<script type="text/javascript">
  Ext.Ajax.defaultHeaders = { 'X-CSRFToken': '1bXpv0Z77idYaU9I7USQLYw3QITi1CGB' };
  Ext.BLANK_IMAGE_URL = "/static/geonode/externals/ext/resources/images/default/s.gif";
</script>

<script src="/static/geonode/script/ux.js"></script>
<script src="/static/geonode/script/GeoNode.js"></script>

<link                              rel="stylesheet"                              type="text/css"
href="/static/geonode/externals/openlayers/theme/default/style.css" />
<script src="/static/geonode/script/OpenLayers.js"></script>
<link  rel="stylesheet"  type="text/css"  href="/static/geonode/externals/geoext/resources/css/geoext-
all.css" />
<script src="/static/geonode/script/GeoExt.js"></script>
<link rel="stylesheet" href="/static/geonode/externals/gxp/theme/all.css" />
<script src="/static/geonode/script/gxp.js"></script>
<link rel="stylesheet" type="text/css" href="/static/geonode/theme/app/geoexplorer.css" />
<!--[if                    IE]><link                    rel="stylesheet"                    type="text/css"
href="/static/geonode/theme/app/ie.css"/><![endif]-->
<script src="/static/geonode/script/GeoExplorer.js"></script>
<script type="text/javascript">
OpenLayers.ImgPath = "/static/geonode/externals/openlayers/img/";
</script>

<script
src="http://www.google.com/jsapi?key=http://maps.google.com/maps/api/js?v=3.7&amp;amp;senso
r=false"></script>
<script type="text/javascript">
   google.load("earth", "1");
</script>

<link  rel="stylesheet"  type="text/css"  href="/static/geonode/theme/ux/colorpicker/color-
picker.ux.css" />
```

```
<style type="text/css">
   #templates { display: none; }
</style>

<script src="/static/geonode/script/PrintPreview.js"></script>
<link rel="stylesheet" type="text/css" href="/static/geonode/externals/PrintPreview/resources/css/printpreview.css" />
<script src="http://192.168.0.164/geoserver/pdf/info.json?var=printCapabilities" type="text/javascript"></script>

    <link rel="stylesheet" type="text/css" href="/static/theme/site.css" />
    <script type="text/javascript" src="/jsi18n/"></script>
    <script type="text/javascript" src="/lang.js"></script>


<script type="text/javascript">
var app;
Ext.onReady(function() {

    var config = Ext.apply({
        tools: [{
            ptype: "gxp_wmsgetfeatureinfo",
            // comment the line below if you do not want feature info in a grid
            format: "grid",
            actionTarget: "main.tbar",
            outputConfig: {width: 400, height: 300}
        }],
        proxy: "/proxy/?url=",

        /* The URL to a REST map configuration service.  This service
         * provides listing and, with an authenticated user, saving of
         * maps on the server for sharing and editing.
         */
        rest: "/maps/",
        homeUrl: "/",
        localGeoServerBaseUrl: "http://192.168.0.164/geoserver/",
        localCSWBaseUrl: "http://192.168.0.164/geonetwork/srv/en/csw",
        csrfToken: "1bXpv0Z77idYaU9I7USQLYw3QITi1CGB",
        authorizedRoles: "ROLE_ADMINISTRATOR"
    }, {"defaultSourceType": "gxp_wmscsource", "about": {"abstract": "", "title": "Poste De Patrouille Map"}, "map": {"layers": [{"opacity": 1.0, "group": "background", "name": "ROADMAP", "title": "Google Roadmap", "selected": false, "visibility": true, "source": "0", "fixed": true}, {"opacity": 1.0, "group": "background", "name": "TERRAIN", "title": "Google Terrain", "selected": false, "visibility": false, "source": "0", "fixed": true}, {"opacity": 1.0, "args": ["No background"], "group": "background", "name": "No background", "title": "No background", "selected": false, "visibility": false, "source": "1", "fixed": true, "type": "OpenLayers.Layer"}, {"opacity": 1.0, "args": ["OpenStreetMap"], "group": "background", "name": "OpenStreetMap", "title": "OpenStreetMap", "selected": false, "visibility": false, "source": "1", "fixed": true, "type": "OpenLayers.Layer.OSM"}, {"opacity": 1.0, "group": "background", "name": "osm", "title": "MapQuest OpenStreetMap",
```

"selected": false, "visibility": false, "source": "2", "fixed": true}, {"opacity": 1.0, "group": "background", "name": "naip", "title": "MapQuest Imagery", "selected": false, "visibility": false, "source": "2", "fixed": true}, {"opacity": 1.0, "group": "background", "name": "AerialWithLabels", "title": "Bing Aerial With Labels", "selected": false, "visibility": false, "source": "3", "fixed": true}, {"opacity": 1.0, "styles": "poste_de_patrouille_layer", "name": "geonode:poste_de_patrouille_layer", "format": "image/png", "cached": true, "selected": true, "visibility": true, "capability": {"abstract": "This layer shows the distribution of ranger posts.", "nestedLayers": [], "cascaded": 0, "fixedHeight": 0, "prefix": "geonode", "keywords": ["ranger"], "noSubsets": false, "dimensions": {}, "opaque": false, "tileSets": [{"layers": "geonode:poste_de_patrouille_layer", "styles": "", "format": "image/jpeg", "height": 256, "srs": {"EPSG:900913": true}, "bbox": {"EPSG:900913": {"srs": "EPSG:900913", "bbox": [-20037508.34, -20037508.34, 20037508.34, 20037508.34]}}, "resolutions": [156543.03390625, 78271.516953125, 39135.7584765625, 19567.87923828125, 9783.939619140625, 4891.9698095703125, 2445.9849047851562, 1222.9924523925781, 611.4962261962891, 305.74811309814453, 152.87405654907226, 76.43702827453613, 38.218514137268066, 19.109257068634033, 9.554628534317017, 4.777314267158508, 2.388657133579254, 1.194328566789627, 0.5971642833948135, 0.29858214169740677, 0.14929107084870338, 0.07464553542435169, 0.037322767712175846, 0.018661383856087923, 0.009330691928043961, 0.004665345964021981, 0.0023326729820109904, 0.0011663364910054952, 0.0005831682455027476, 0.0002915841227513738, 0.0001457920613756869], "width": 256}], "infoFormats": ["text/plain", "application/vnd.ogc.gml", "text/html"], "styles": [{"abstract": "", "title": "", "legend": {"height": "20", "width": "20", "href": "http://192.168.0.164:80/geoserver/wms?request=GetLegendGraphic&format=image%2Fpng&width=20&height=20&layer=poste_de_patrouille_layer", "format": "image/png"}, "name": "poste_de_patrouille_layer"}], "attribution": {"title": "gis"}, "authorityURLs": {}, "bbox": {"EPSG:32735": {"srs": "EPSG:32735", "bbox": [724930.205717801, 9823061.7029768, 827916.855017346, 10103164.5631972]}}, "fixedWidth": 0, "metadataURLs": [{"href": "http://192.168.0.164/geonetwork/srv/en/csw?outputschema=http%3A%2F%2Fwww.isotc211.org%2F2005%2Fgmd&service=CSW&request=GetRecordById&version=2.0.2&elementsetname=full&id=c49078f2-74d7-11e2-9a0e-080027df8e4d", "type": "TC211", "format": "text/xml"}], "name": "geonode:poste_de_patrouille_layer", "identifiers": {}, "srs": {"EPSG:900913": true}, "formats": ["image/png", "application/atom xml", "application/atom+xml", "application/openlayers", "application/pdf", "application/rss xml", "application/rss+xml", "application/vnd.google-earth.kml", "application/vnd.google-earth.kml xml", "application/vnd.google-earth.kml+xml", "application/vnd.google-earth.kml+xml;mode=networklink", "application/vnd.google-earth.kmz", "application/vnd.google-earth.kmz xml", "application/vnd.google-earth.kmz+xml", "application/vnd.google-earth.kmz;mode=networklink", "atom", "image/geotiff", "image/geotiff8", "image/gif", "image/gif;subtype=animated", "image/jpeg", "image/png8", "image/png; mode=8bit", "image/svg", "image/svg xml", "image/svg+xml", "image/tiff", "image/tiff8", "kml", "kmz", "openlayers", "rss"], "title": "Poste De Patrouille Layer", "queryable": true, "llbbox": [29.020971349904563, -1.5998094080682752, 29.946739169795446, 0.9327768107622747]}, "source": "4", "title": "Poste De Patrouille Layer", "fixed": false, "transparent": true}], "center": [3297162.2279277, -29340.768678386], "units": "m", "maxResolution": 156543.03390625, "maxExtent": [-20037508.34, -20037508.34, 20037508.34, 20037508.34], "zoom": 9, "projection": "EPSG:900913"}, "id": 2, "sources": {"1": {"id": "2", "ptype": "gx_olsource"}, "0": {"apikey": "http://maps.google.com/maps/api/js?v=3.7&sensor=false", "id": "1", "ptype": "gxp_googlesource"}, "3": {"id": "4", "ptype": "gxp_bingsource"}, "2": {"id": "3", "ptype": "gxp_mapquestsource"}, "4": {"title": "My GeoServer WMS", "url": "http://192.168.0.164/geoserver/wms", "baseParams": {"VERSION": "1.1.1", "REQUEST":

```
"GetCapabilities", "TILED": true, "SERVICE": "WMS"}, "ptype": "gxp_wmscsource", "restUrl": "/gs/rest", "id": "0"}}});

   app = new GeoExplorer(config);

   var permalinkTemplate = new Ext.Template("{protocol}//{host}/maps/{id}");
   var permalink = function(id) {
         return permalinkTemplate.apply({
            protocol: window.location.protocol,
            host: window.location.host,
            id: id
         })
      };

         var moreInfoTemplate = new Ext.Template(decodeURIComponent(Ext.get("more-info-tpl").dom.innerHTML));
   var mapInfoHtml = config.id ? moreInfoTemplate.apply({permalink : permalink(app.mapID)}) : "This map is currently unsaved";
   Ext.DomHelper.overwrite(Ext.get("more-info"), mapInfoHtml)

   var titleTemplate = new Ext.Template(Ext.get("title-tpl").dom.innerHTML);
   Ext.DomHelper.overwrite(Ext.get("map-title-header"), titleTemplate.apply({title: config.about.title}));

   app.on("saved", function(id) {
      //reset title header
      Ext.DomHelper.overwrite(Ext.get("map-title-header"), titleTemplate.apply({title: config.about.title}))

      //reset more info link
      Ext.DomHelper.overwrite(Ext.get("more-info"), moreInfoTemplate.apply({permalink : permalink(app.mapID)}))
   }, this);

});
</script>


</head>
<body>

<div id="header-wrapper">

   <div id="header">
     <div class="wrap selfclear">
       <a id="logo" href="/">GeoNode</a>
       <div id="login-area">
```

```
        <a href="/profiles/gis/">gis</a> (<a href="/accounts/password/change/">Change password</a> | <a href="/accounts/logout/">Log out</a>)

      </div>
    </div><!-- /.wrap -->
  </div><!-- /#header -->

 <div id="topPanel">
  <div id="more-info"></div>
  <span id="map-title-header"></span>
 </div>
</div>
<div id="templates">
 <div id="more-info-tpl"><a class='link' href='{permalink}'>View info</a></div>
 <div id="title-tpl"><a class='maplist' href='/maps/search'>Maps</a> / <strong>{title}</strong></div>
</div>

</body>
</html>
```