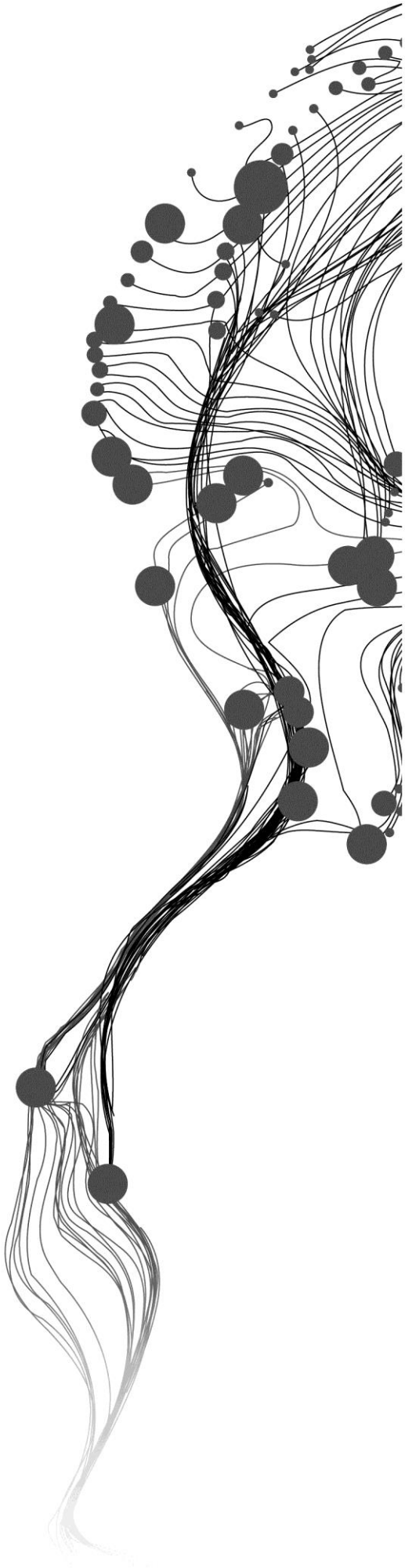
WEB ARCHITECTURE FOR THE PROVISION OF VISUALIZATION AS A SERVICE (VAAS)

SAJID ANWAR

April, 2014

SUPERVISORS:

Dr.J.M.Morales
B.J.Kobben,M.Sc



WEB ARCHITECTURE FOR THE PROVISION OF VISUALIZATION AS A SERVICE (VAAS)

SAJID ANWAR

Enschede, The Netherlands, April, 2014

Thesis submitted to the Faculty of Geo-Information Science and Earth Observation of the University of Twente in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation.
Specialization: GFM

SUPERVISORS:
Dr.J.M.Morales
B.J.Kobben,M.Sc

THESIS ASSESSMENT BOARD:

Prof.Dr. M.J. Kraak (Chair)
Dr. O. Huisman,(External examiner, Rosen Inspection)

ABSTRACT

Traditionally, most of the visualization applications are designed for organization specific-applications and can render only specific dataset confined to that application. Moreover, these systems are unable to handle arbitrary dataset to create other kind of visualization, and these visualization applications cannot be used by application developers. This make these systems to confine to a specific dataset and hence to that visualizations. This study envisage the use of Visualization as a Service (VaaS), to prepare semi-automate choropleth visualizations from arbitrary spatial dataset on demand, and to expose their operations through service application programming interface (API), to the users, that could be further used by application developers in their applications. To achieve this target, we presented the results of experiments with the design of algorithms for capturing existing choropleth design rules, and their role in service classes used for the construction of web service that take data classification and produce rules of visualizations used for data interpretation (e.g. choropleth maps). Attempt was also made to develop loosely coupled visualization framework for the provision of visualization service. This study reveals that Vaas can be effectively used to develop semi-automated visualizations from arbitrary dataset on demand, and enhance cartographic visualizations functionality. The Interface is exposed to the user and therefore can be consumed by a client application.

Keywords:
Cartography, Web Service, SOAP, REST, Map file, WMS, MapServer

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the help and support of Allah, the Most Gracious, and the Most Merciful.

Special thanks to my parents. Words cannot express how grateful I am to my parents for all of the sacrifices that you've made on my behalf.

Next, I would like to pass my deepest gratitude to my supervisors Dr. Javier Morales and Drs. B.J. Barend Kobben for their useful comments, excellent guidance and remarks all the way through the learning process.

I would like to thank the Royal Netherlands Government for funding this study through the Netherlands Fellowship Program. I will never and ever forget the input and knowledge I got from the staff members at ITC, University of Twente.

During my stay in ITC, I had the opportunity to meet many colleagues from many different countries. But I would like to thank my friends Saad Khan, Nayyer Saleem, Dr. Amjad Ali, Dr. Saleem Ullah, Birane Dia, Zahid Ali, Qasim Khan and Hazrat Hussain. Brothers thanks for your company and advices all over one and half year especially when I was in doubt. I would like to express my deepest gratitude to my brother Sadiq Anwar and my beloved kids Hooria Jan and Muhammad Hamdan. It was really painful missing you for the past one and half a year. You are the reason for my strength, without your encouragement and support it would have been be hard to reach here.

At the end, I would like express appreciation to my beloved wife for her support and sacrifices while I was away. Thanks also to all my sisters and brothers.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

API	Application Programming Interface
CGI	Common Gateway Interface
DBMS	Database Management System
GUI	Graphical User Interface
Java EE	Java Platform, Enterprise Edition
JAX-WS	Java API for XML Web Service
JSP	Java Server Pages
JPEG	Joint Photographic Experts Group
MS4W	MapServer 4 window
QGIS	Quantum Geographical Information System
REST	Representational state transfer (
VaaS	Visualization as a Service
WMS	Web Map Service
XML	Extensible Markup Language
SRID	Spatial Reference System Identifier
IDE	Integrated Development Environment
SOAP	Simple Object Access Protocol
WSDL	Web Service Definition language
OGC	Open Geospatial Consortium
pyDev	Python IDE for Eclipse
HTTP	Hyper Text Transfer Protocol
UDDI	Universal Description, Discovery and Integration
WWW	World Wide Web
3WC	World Wide Web Consortium

1. INTRODUCTION

This chapter introduces context of this study, its motivation and the driving factors. The focus lies on finding the possibility of providing visualization as a service, a solution towards a semi-automatic generation of choropleth visualizations from arbitrary datasets. The service will also provide interface to expose their operations to the user who adopts them in other applications.
This chapter is organized in five sections. Section 1.1 describes motivation for the research. Section 1.2 identifies the research problem by presenting a brief overview of the available visualizations applications and their limitations, thereby providing insights to design loosely coupled web service for these visualization applications. Section 1.3 presents the main objective, and sub-objectives of this study, while section 1.4 outlines the research questions to be answered. Finally, section 1.5 provides a brief structure of this thesis.

1.1 Motivation for the study

Besides providing an interface for communication and data dissemination, the web provides an important medium for concurrent distribution of user-interactive geospatial data which are both platform- and location independent (Cerba & Cepicky, 2012). Due to the wide spread availability of open standard web services, and open data repositories, it is now possible to bring together applications as a package of service distributed over the web. These packages of services produce results that still need proper presentation to different users and user communities. Usually, the cartography of these data is done manually; however, automatic visualizations of these statistics could be very beneficial for those who have no cartographic expertise (Zeng & Zhao, 2011), because the creation of map poses many challenges and need knowledge of cartography to find and solve the problems (Iosifescu Enescu, 2011).
The process of designing cartographically rich maps is time consuming because such maps must adhere strictly to cartographically correct selection of data, visual variables, color, labelling and layout (Smith, 2010). This implies that correct map generation is the key, as visualization lets us to comprehend and maintain higher amount of information compared to text-based communication. Therefore without correct visual image, recalling the same information would be difficult unless an extensive list of area descriptions is adequately captured (Dobesova & Brus, 2012).

The widespread availably of GIS technologies and cartographic software makes it possible to create maps without high level of expertise. But these software do not provide guidance for the creation of cartographically correct maps, such that the produced map may be of low quality (Smith, 2010). If the cartographer's knowledge is harnessed and reshaped in a visualization service, a user with any level of mapping skills can use and benefit from it (Smith, 2010). This minimizes the problems of unsatisfactory knowledge of cartographic rules and open the door to generate semi-automatic maps by non-cartographers, through various techniques; one of which is the design of web service to provide visualization facilities to large communities (Jan, Zdena, Jaromir, & Vilem, 2010).

Some organizations demand the production of real-time online map visualizations. To present the latest view of these data, maps are required to reflect the current situations immediately without the violations of cartographic rules. Therefore we need a system which provides automatic or semantic approach for the rapid visualization development over the web to offer online services independent of physical geography of the users. Consequently, we need a service which provides cartographically correct visualizations on demand.

1.2 Problem Identification

Traditionally, most of the visualization applications are designed within the boundaries of enterprises i.e. maps are produce for organization specific-applications. For example NANOOS Visualization System (Risien et al., 2009), ,Thematic Mapping Services (Lili, Qingwen, & An, 2010) and web-based visual platform to access climatological fields from model (Sun et al., 2012).These services are designed for specific application and can render only specific dataset confined to that application. Moreover, these systems are unable to handle arbitrary dataset to create various types of visualizations as shown in Figure 1-1, which shows various dataset, and the rendering of choropleth visualizations. For each type of visualization, there is a separate application system. The same application system cannot use for other dataset to produce other visualizations. Therefore these systems are confined to specific dataset and hence to that visualizations. Furthermore, these visualization applications cannot be used by application developer in their tasks, because the architecture of these applications does not offer any interface for their intended functionality. An interface is a gateway to interact with the system and use their services without knowing their internal complexity.

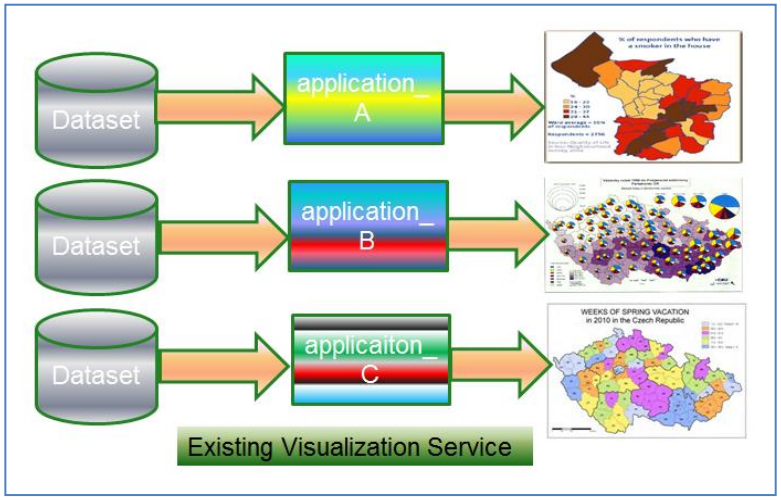


Figure 1-1 Tightly couple visualization service, each map has a separate application process

The concepts of proposed loosely couple visualization service is shown in Figure 1-2, where arbitrary datasets could fit into the service. The service will be capable of handling these dataset and process them semi-automatically to create various types of visualizations, based on the measurement scale of the data. In the middle layer service comprises of cartographic rules. That are structured in computer application programs which determines visual variables like *colour*, *value* for data measurement scale i.e. ratio, ordinal, along with automatic classification for a cartographically correct choropleth visualizations.

The main task of this service is to create cartographically correct semi-automatic choropleth visualizations from arbitrary dataset, and to expose their operations through service application programming interface (API), to the users, that could be further used by application developers in their own application. API's are components that have a published interface by which third-party components may interact with them (Alonso, Casati, Kuno, & Machiraju, 2004). For the purpose in short, this service will be called Visualization as a Service (VaaS) to provide loosely coupled visualizations.

In this research, we assume that spatial dataset following Esri shapefile standard are available and stored in spatial database, using any open source database management system such as PostgreSQL. We also assume that user know their data measurement scale in advance, and should submit this choice along with data attribute for visualizations and classifications.

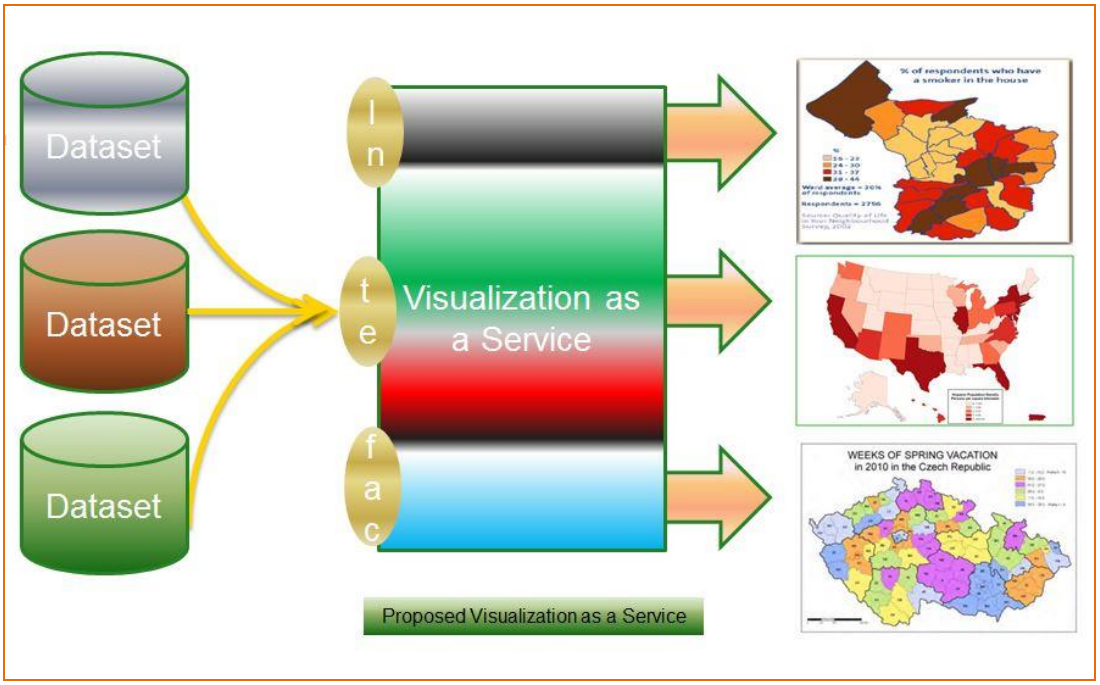


Figure 1-2 Proposed semi-automatic loosely coupled visualization service

1.3 Objective

1.3.1 General objective

To develop loosely coupled visualization framework for the provision of Visualization as a Service (VaaS).

The following sub-objectives have been put forward to address the main research objective above:

1.3.2 Sub-objectives

1. To properly define Visualization as a Service (VaaS) within the context of this research project;
2. To study existing techniques or specification of visualization and explore their possibilities as a service, and to make it part of existing web architectures;
3. To develop an interface (i.e. web service), that wraps around visualization services and allow data service or methods to be associated loosely;
4. To develop a framework for visualization services, such that a service user can understand the functionality of the service and how to access this service;
5. To decide which technology could be used to realize these implementations; and
6. To use a case study of a choropleth visualization to test the VaaS

1.3.3 Research Questions

Research questions to be answer are list below:

Related with objective 1:

a) What is the definition of VAAS within the context of this research?

Related with objective 2:

a) Which techniques or specification to use as component of the visualization system?

Related with objective 3 & 5:

a) How to associate data service with the database to retrieve arbitrary dataset?
b) Which dataset to use for visualization?
c) How to formulate rules for different data measurement level?
d) How to develop web service that wraps around visualizations service or methods?
e) How to structure visualization rules produce by the different web service methods?
f) Which technology to use for web service creation?

Related with objective 4 & 5:

a) How to develop map file for each data measurement level?
b) How to read rules from the service to create map file?
c) How to design Web Map Service request in mapfile?
d) Which engine to use for rendering?
e) How to associate all components of VaaS within a framework?
f) Which tools and technology to use for map file creation?

To *Related with objective 5 & 6:*

a) Which choropleth visualizations to use to test the service?
b) Which base map to use to overlay visualizations?
c) How to overlay arbitrary visualizations on base map?
d) Which tool and technology to use for service GUI?

1.3.4 Innovations Aimed At

The innovation of this research project is designing a visualization web service that provides visualizations from arbitrary dataset. This new concept of visualization as a service (VAAS) has not been introduced in previous literature.

1.4 Structure Of The Thesis

Chapter 1: Introduces the research gap, explains the research problem, objectives and questions for the development of a semi-automated service for choropleth visualizations derived from arbitrary datasets.

Chapter 2: This is a review of literature on two broad concepts of web service and map server, which are basically the existing concepts and technologies for web-based map generation.

Chapter 3: Includes an overview of thematic cartography, a description of measurement levels of data and of visual variables. The chapter also covers classification approaches such as equal interval steps-; and harmonic series.

Chapter 4: This chapter examines design of cartographic visualization algorithms comprising classification and assignment of colour on the basis of data measurement levels.

Chapter 5: This chapter discuss creation of Map files and combination of this Map file with other components of VaaS leading to an implementation framework for Choropleth visualization.

Chapter 6: This chapter examines the implementation of VaaS using a combination of tools and technologies which helps to combine arbitrary datasets for cartographically correct choropleth visualization.

Chapter 7: This chapter provides a summary of visualization results, as well as conclusions and recommendations.

2. REVIEWS OF WEB SERVICES AND MAPFILE

This chapter examines two broad concepts of web service and mapserver map file. Section 2.1 commences with the definitions of web service from different perspectives; Section 2.2 outlines some advantage of web service; Section 2.3 describes two main types of web services i.e. SOAP and REST base services. Section 2.4 gives a brief overview of cartographic web service comprising OGC WMS. A review of literature on map server and map file is provided in section 2.5, while the literature review is concluded with a reflection in section 2.6.

2.1 What are Web Services

Web services are in use since last decade (Alonso et al., 2004). Many software vendors and organizations are involved in the development of web service applications and standards. However, in early days of Web service development, due to lake of common understanding there was no single, recognized definition of web services (Joshua, 2005).

A Web service, in very broad term, is a method of communication between two applications programs over the World Wide Web (WWW) (Sandoval, 2009). Also can be stated as, a service which is always available over the network to other software components, and can access from anywhere, anytime without understanding their internal complexity (Joshua, 2005). According to World Wide Web Consortium (W3C), Web service can be defined as a software system designed to provide interoperability between web applications or machines that are distributed over the network such as Internet. These services expose their operations through a well-defined Interface. Other services use this interface to make connection with the system in a prescribed manner, and conveyed information using Hyper Text Mark-up Languages (HTTP)[1](Fielding et al., 1999), with Extensible Mark-up Languages (XML) [2](W3C) serialization. Universal Description, Discovery and Integration (UDDI)[3], protocol are used (Wikipedia, 2013b), to find service dynamically. Most of the Web services follow the principles of software modularity, self-expression capabilities (Michael, 2012).

Web Service follows the basic concept of Web communication (Alonso et al., 2004). It consists of request-response pattern as shown in Figure 2-1. In this scenario the service provider provide the available services on behalf of consumer request. It is a two way process, the service is request from the client side and in return service provider transferred the requested service if available.

Figure 2-1 Web Service request-response mechanism

[1] http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
[2] http://en.wikipedia.org/wiki/XML
[3] http://en.wikipedia.org/wiki/Universal_Description_Discovery_and_Integration

2.2 Benefit of Web Services

Web services provide interoperability among different applications (Alonso et al., 2004). The goals of interoperability is to provide seamless and automatic connections from one software application to another (Cohen, 2002). For example in case of SOAP based (discussed later) web services SOAP[4],WSDL [5]and UDDI protocols define self-describing facility to discover and call a method in a software application regardless of location and platform (Michael, 2012). The XML technologies and HTTP can be used by the user to communicate with web services(ORACLE, 2013). The client only needs to know about the WSDL definition to successfully connect with Web services and exchange the data or messages(Cohen, 2002). Hence, interoperability result in different vendors software interaction across local or wide area networks possible.

Versatility is another benefit of Web services. Web services are accessible by different consumers such humans via a Web-based client interface or by other Web services through programming interfaces (Cohen, 2002). A client can combine data from multiple Web services, which are distributary located (Alonso et al., 2004). Moreover, as communication is through Web services any change in the underlying database will not affect the web service itself (Michael, 2012).

2.3 Types of Web Services

On technical basis Web services can be implementing in various ways.Pautasso, Zimmermann, and Leymann (2008) and Michael (2012) describe two types of Web services models;

- Simple Object Access Protocol (SOAP)
- Representational State Transfer (REST)

2.3.1 SOAP/WSDL-base Web Services

SOAP defines a standard communication protocol specification for XML-based message exchange, using various transport protocols, such as HTTP and SMTP (Allamaraju, 2010).SOAP web service based on WSDL[6]/SOAP standards (Michael, 2012). SOAP is XML based communication protocol for exchange of messages among disparate computers, irrespective of their operating systems platforms, and programming languages (Alonso et al., 2004).

WSDL is used to describe the interface of Web services, and exposed on the Internet, where other applications interact with this interface without knowing their internal complexity of implementations (Pautasso et al., 2008). For the interaction with the service also known as service consumption, client makes use of WSDL also called WSDL file (Alonso et al., 2004). WSDL offer a machine readable descriptions of how the service can be invoked, their parameters requirement, and formats of returns message (Michael, 2012). For the message exchange SOAP are used, where the capacity of SOAP exchange operations broader than REST (NetBeans, 2013a).

[4] http://en.wikipedia.org/wiki/SOAP
[5] http://www.w3.org/TR/wsdl

The suitability of SOAP-based web services are more appropriate in applications with complex operations, sophisticated security and communication reliability are demanding (Michael, 2012). SOAP base service also suitable in case where other communication protocol is needed is instead of HTTP (NetBeans, 2013a).

SOAP is a standard message protocol used to exchange XML based message over the HTTP for accessing Web services (Michael, 2012). They codify XML as encoding schemes for request and response parameter over the HTTP. Figure: 2-2, show communication between server requester and server provider, where SOAP message are exchange on top of not only HTTP but also SMTP or other transport protocol.

Java API for XML Web service (JAX-WS) are used as model for SOAP –base web services using Java development environment (NetBeans, 2013a). In JAX-WS annotation are available to develop web service and their consumer i.e. clients.

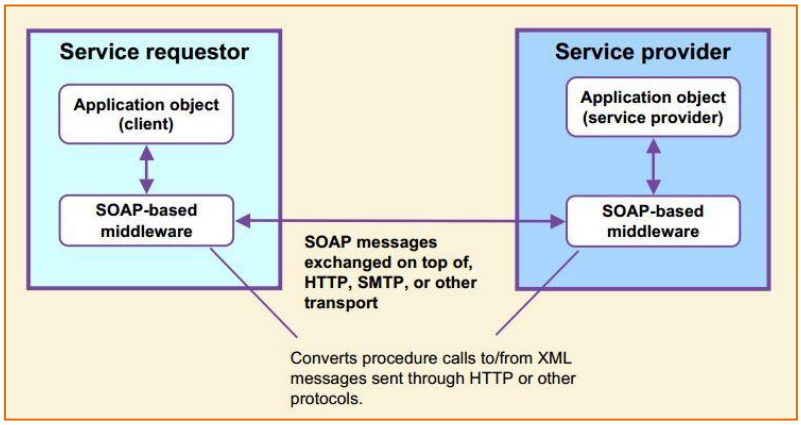


Figure 2-2 SOAP message between service requester and provider (Michael, 2012)

For the consumption of web service we need to develop clients, such as in case of Java development environment we create a servlet class and web page as clients of service, where user pass information from one client i.e. web page to servlet (NetBeans, 2013a). The servlet is java programming class used to extend the capability of server and provide bridge between html client and server (Michael, 2012).

Using NetBeans IDE for the creation of JAX-WS client is synchronous by default. In case of synchronous communication client invoke a request on a service and then suspend their processing until receive response from server (Alonso et al., 2004). While in asynchronous communication client continue processing without waiting for response from server side.

Asynchronous client consume web service using two way approach, i.e. polling or call-back approach (NetBeans, 2013a). In case of polling approach we called web service method and repeatedly ask for the response, result in blocking operations such calling threads (NetBeans, 2013a). While using 'callback' approach we send call message to third method, and they communicate with server and provide the result to the client, and hence no need to wait.

Using NetBeans IDE, we can add support for asynchronous application with only ticking a box in one of the available tool as shown in Figure 2.4 (NetBeans, 2013a). By enabling asynchronous client option, we can include the above approach in our clients as shown in Figure 2.3. Both synchronous and asynchronous options are available. We can only in our clients these methods with drag and drop.

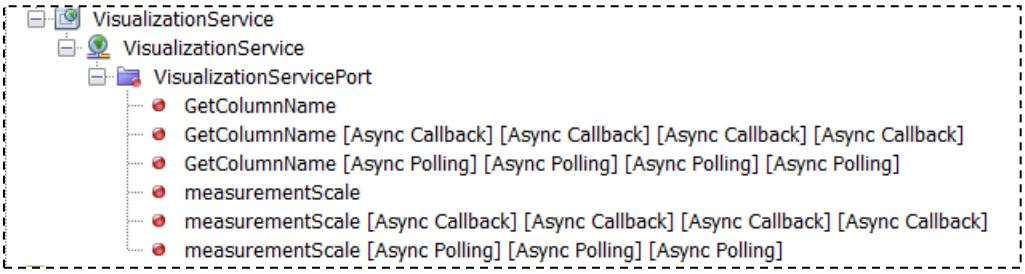


Figure 2-3 Web service clients synchronous and asynchronous approach using Java NetBean IDE

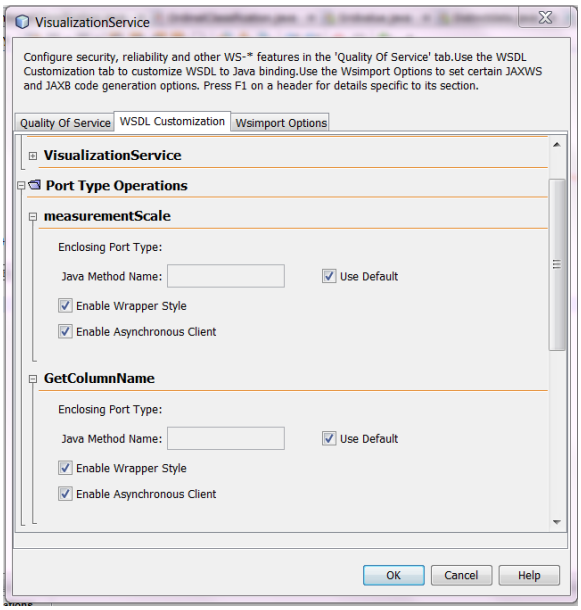


Figure 2-4 WSDL customization for asynchronous Web service clients

2.3.2 REST-base or Restful Web Services

REST stands for REpresentational State Transfer was coined by Roy Fielding, in his doctoral dissertation (Allamaraju, 2010). REST is neither a protocol nor standard but is simply an architecture style like other styles, for example client-server architecture. Therefore Web services which follow REST type of styling are called Restful Web services (Vinoski, 2008).

- Resource

A RESTful resources is something addressable over the Internet (Sandoval, 2009). By addressable we mean during client server communication resources are accessible. A resource may be temperature in enschede at 3:00 pm EST, visualization of nominal data etc. REST is a resource centre approach as compared to message centric SOAP (Pautasso et al., 2008). Hence every document and process is declared as a web resource and identified using URI (discussed later) (NetBeans, 2013b). The management of these resources achieved through actions stated in HTTP header (Allamaraju, 2010). Message exchange can be done with any format such as, xml, json or html etc., and neither SOAP nor WSDL are used (NetBeans, 2013b).For example, Web page is a resource of Website having unique URL. And representation is the HTML document communicated between client and server. Representation present resource

encapsulation information for example state and data which are encoded in XML or JSON (Allamaraju, 2010).

- Representation

Representation of resources in the context of Restful web service is the sending materials between clients and servers (Sandoval, 2009). It is a temporal state of real data placed in some storage once requested (Allamaraju, 2010). In life of a web service various clients can request resources. The same resource can be consumed with different representation for different client request. Hence various type of representation possible of the same resources, using the same url such as a text, file, xml, json (Sandoval, 2009). Example of automate request representation where readability is not concern for user is the XML. While for human generated request using browser is the html page.

- URI

A Uniform resource Identifier, or URI, in the context of RESTful web service is a hyperlink of a resource, provide means to exchange representation between client and server (Allamaraju, 2010). They are used for the unique identification of resource (Allamaraju, 2010). The resources are conceptually separate from the representation returned to the client, for example database server send some XML, JSON[7] or HTML instead of database (Roger, 2008).

- Uniform interface through HTTP request

In modern web application development to limit the design and implementation ambiguity, actions on resources are limit to CRUD i.e. Create, Retrieve, Update and Delete (Sandoval, 2009). HTTP a "a protocol that allows for sending documents back and forth on the web"(Sandoval, 2009) make use of POST,GET,PUT and DELETE methods.

Sandoval (2009), compare RESTful CRUD actions, to that of HTTP methods as under.

The method GET is used to retrieve the resources. For example to retrieve XML representation of data classification classes (discussed in chapter 3), with location http://restfulvaas.com/. The representations of classes are shown in Table 2.2, comprise of parent element <Map>, and child element <classes>.

Table 2-1 REST CRUD vs HTTP methods

Data action	HTTP protocol equivalent
CREATE	POST
RETRIEVE	GET
UPDATE	PUT
DELETE	DELETE

[7] http://json.org/

Table 2-2 XML representation of data classes

```
<Map>
<classes>
< Id>1 </Id>
 <minBoundary>83.0</minBoundary>
 <maxBoundary>66.4</maxBoundary>
<red> 240</red>
<green> 48</green>
<blue> 48</blue>
</classes>
<classes>
< Id>2 </Id>
 <minBoundary>66.4</minBoundary>
 <maxBoundary>49.8</maxBoundary>
<red> 237</red>
<green> 96</green>
<blue> 96</blue>
</classes>
<classes >
< Id>3</Id>
 <minBoundary>49.8</minBoundary>
 <maxBoundary>33.2</maxBoundary>
<red> 234</red>
<green> 144</green>
<blue> 144</blue>
</classes>
</Map>
```

To define URI of our define representation, as example of the following

http://restfulvaas.com/Map to access a list of classes, and to retrieve specific class we can used the following URI; to access specific classes with id.

http://restfulvaas.com/Map/classses/ id.

To make a request to web service consist of the above data classification records, and we are interested in class with id=1, the URI of request will be http://restfulvaas.com/Map/classses/1. The response from the server may look like:

```
<classes>
< Id>1 </Id>
 <minBoundary>83.0</minBoundary>
 <maxBoundary>66.4</maxBoundary>
<red> 240</red>
<green> 48</green>
<blue> 48</blue>
</classes>
```

In case we are interested in all the records we can used the URI http://restfulvaas.com/Map/classess, the response may look like in Table 2.2.

In the above request and response session, we (clients) make an HTTP request with GET method and '1' as an identifier for the class. The client send the representation type let say xml using the header of HTTP, the web server interpret the GET request and pass to the RESTful framework, and after finding the records, forward to the server and coverts to user requested format. The information is transferred to the client using HTTP response. For the description of others methods See (Sandoval, 2009).

To summarize the main features of are as follow (Roger, 2008):

- REST is used across platform due to their work relation with HTTP.
- REST is simple because they are using XML technology instead of heavyweight SOAP/XML.
- REST is Stateless, no need to remember the links. As for stateless messaging style all the required information are included in each request to interpret the message correctly.
- REST has uniform interface, as clients use common GET and POST method of HTTP, for communication. Moreover, uniform interface result in self-describing communication between client and server(Allamaraju, 2010).

In this research work we have interest to use both SOAP and REST architectural style for development of VaaS, but more emphasis on SOAP and REST is depend on the availability of time period.

2.4 Web Services in cartography

The Web an important medium for geospatial data distribution, provide platform independent environment, where many users can interact with the maps concurrently from distributed locations, result in location independent (Cerba & Cepicky, 2012). These maps are called web mapping, ease to interact and update.

Web mapping is the method of designing, implementing, producing and distributing maps on the world Wide Web (WWW) (Rautenbach, Coetzee, & Iwaniak, 2013).

The usage of the web as a distribution medium for geospatial data revolutionize web cartography, and opens new opportunities in data visualization, such interactive maps.

For geospatial data visualization and processing, most of services are standardized by Open Geospatial Consortium (OGC)[8], called OGC service specifications. Various service specifications are frequently used in cartography. In this research work we used Web Map Service (WMS) , to produce maps from spatial dataset. The result maps are pectoral format such as PNG(Portable Network Graphic), GIF (Graphic Interchange Format) etc.(Cerba & Cepicky, 2012).

'The OpenGIS Web Map Service interface standard (WMS) provide a simple HTTP interface for requesting geo-registered map images form one or more distributed geospatial database" (OGC, 2006).

2.5 Review on mapserver mapfile concept

This section reviews the concepts of Mapserver and mapfile.

[8] http://www.opengeospatial.org/

- Mapserver

Mckenna, Fawcett, and Butler (Un-dated), describe mapserver as an open source project that provide dynamic spatial maps rendering to user over the Internet. Mapserver are used as web mapping tools due to their numerous features, such as due to their mapping information availability over the Internet for broad access (Mitchell, 2005). Other reason for their selection is the availability of diverse data formats libraries such as GDAL/OGR, that access different data format without data alteration (Mitchell, 2005).

The main feature of Mapserver describe by (Mckenna et al., Un-dated) are as under:

- Support for various formats such vector, raster and database
- Projections management on the fly
- Support to run on various operation systems
- Best quality rendering
- Support for ready to use open source application environment
- Support for scripting languages and development environment such Pyhton,PHP,JAVA,.Net

Mapserver can be used as a common gateway interface (CGI) application or scripted (API) using web programming languages such Python, PHP etc. (Mitchell, 2005).

"Mapserver works behind the web server application" (Mitchell, 2005). The map requests are received by web server and then pass to mapserver. Mapserver process the request and generates map image as requested, then handover to map server, which is responsible for transmission to user (Mitchell, 2005) as shown in Figure 2-5, where map server is requested, they pass information to mapserver program. The mapserver program search data from various and then pass the information to web server, which result in visualization of map in image format.
Mapserver main task to read data from various source and groups into graphic file known as map image (Mckenna et al., Un-dated). Different layers are overlaid on top of other render in user friendly format to visualize (Mitchell, 2005).

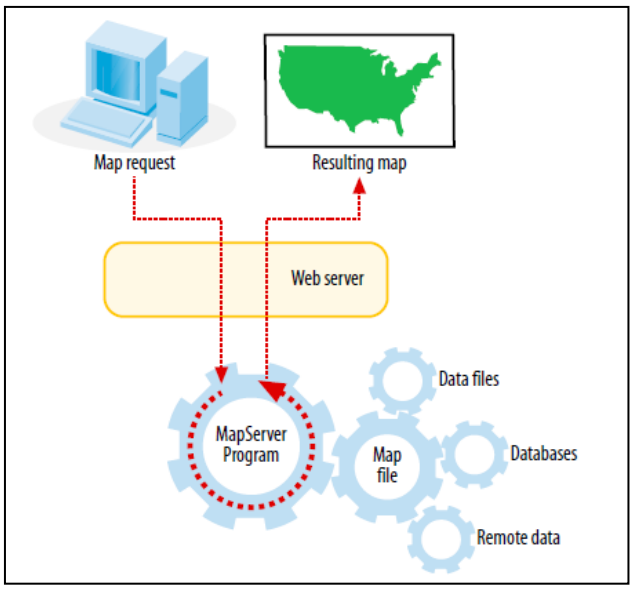


Figure 2-5 A diagram showing the basic operation of a mapserver application (Mitchell, 2005)

- Main components of Mapserver

The main components of mapserver includes CGI program, map file, data source and output images (Mitchell, 2005), also shown in Figure 2-6, where users make request and CGI program of mapserver access map file, act according to the available instructions, retrieve information from the data source and return image map.

CGI executable application run on web server is the simplest form of Mapserver. CGI program receive request from web server, process and return the result for further information (Mitchell, 2005).Map file is text configuration file that composed of information setting for map drawing and interaction (Mitchell, 2005). Information tells the Mapserver what to draw, how to draw, and where data is located.

A simple structure of mapserver map file is shown in Figure 2-7; consist of one layer countries, of 'polygon' type.

Mapserver application request must specify map file to use. The mapserver act according to the content of map file, and render the map. Therefore map file is the core part of any mapserver application (Mitchell, 2005).

Data source is another component of map server; these may be GIS data files such shapefile, spatial database connection.

Output map image is the final outcome of all the above components and process. This component is the actual output, user interested and render in client browser. Various format are support such GIF, PNG, JPEG etc.

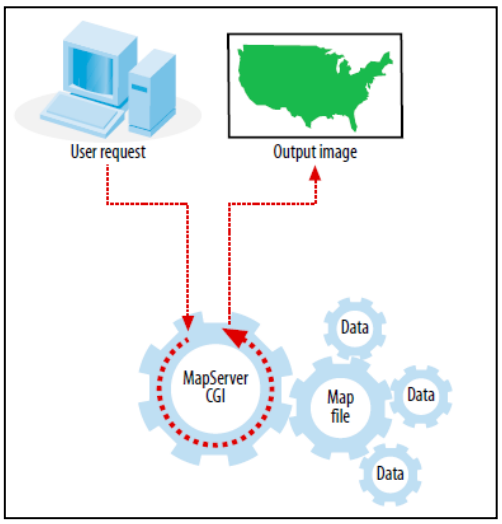


Figure 2-6 The MapScript API's hierarchical object structure (Mitchell, 2005)

```
MAP
  SIZE 600 300
  EXTENT -180 -90 180 90
  LAYER
    NAME countries
    TYPE POLYGON
    STATUS DEFAULT
    DATA countries.shp
    CLASS
      OUTLINECOLOR 100 100 100
    END
  END
END
```

Figure 2-7Basic structure of Mapfile

- MapServer's MapScript

MapServer is used through a web server such as Apache[9] or IIS[10]. We can use mapserver both in CGI or scripting environment. Scripting environment called MapScript (Mitchell, 2005).MapScript provide support for several programming language, such as python, java, php etc. These languages use MapScript API for customize mapping applications.

[9] http://httpd.apache.org/
[10] http://www.iis.net/

According to Mitchell (2005), mapscript expose mapserver functionality to scripting languages to create customize mapping applications; and reduce developing time for programmer.

Mapscript facilitate to load, manipulate, and create map file (Mitchell, 2005).

Mapscript objects

Map file classes can accessed as objects in Mapscript (Mckenna et al., Un-dated; Mitchell, 2005). The main object are *mapObj* consist of *layerObj* and also *legendObj* shown in Figure 2.6. The diagram show objects and how these objects are hierarchically structured.

As shown the main object is the *mapObj*. We can add other objects to it. Some object should never be explicitly create, as these already part of *mapObj*. These objects are *webObj,scaleObj* and l*egendObj*. For further detail (Mckenna et al., Un-dated; Mitchell, 2005).

In this research we are interested to use python or java Mapscript for the creation of map file. Python is more matured as compared to other scripting languages (Gillies, un-dated).

2.6 Concluding remarks

This chapter was comprised of two topic agenda. One section was reserve for web service. And the other part was devoted to mapserver and map file.

Web service is an autonomous software module having self-describing capabilities. They provide interoperability between complexes, heterogonous, distribute systems. Therefore, Web services utilization in the field of cartography for various types of visualization production could play an important role.

Map file is a core part of mapserver, used for spatial data mapping. We discuss Mapscript API, to be used in this research for the creation of map file, to store the decisions generated through visualization web service. The decisions are cartographic rules of various visualizations, the topic of next chapter.

3. THEMATIC CARTOGRAPHY

This chapter presents an overview of thematic cartography, and defines the important of accurate maps. As maps are unique means of communication for spatial information, consequently correctness of transmitted information largely depends upon information visualization on maps. Violations of cartographic rules miss communicate spatial information, and in consequence will mislead map-readers. Therefore, cartographically accurate maps play an important role in decision making, characterized by urgency and criticality.

The chapter is organized as follows: section 3.1 deals with general definition and important of maps in today information age; section 3.2 outline various definition of cartography; section 3.3 deals with thematic cartography and flow diagram of choropleth thematic mapping process; section 3.4 define cartographic data analysis and present overview on measurement scale; section 3.5 describe visual variables and their appropriate usages; section 3.6 define importance of classification and present detail description of equal steps classification approach; section 3.7 deals with choropleth map and their production process; section 3.8 wraps up the chapter by concluding the important of choropleth map, and next step for intelligent implementation of choropleth design rules.

3.1 What is a map?

Due to the current advancement of Geographic Information Science (GIScience) and information technology the significant of map become more versatile (Jan et al., 2010). Maps play a key role in various disciplines including environmental management, urban planning, resources management and education. According to (ICA, 1995), "A map is a symbolized image of geographical reality, representing selected features or characteristics, resulting from the creative effort of its author's execution of choices, and is designed for use when spatial relationships are of primary relevance". A map is a unique tool for sharing and expression of knowledge around the people (Maceachren & Kraak, 1997). A map is a medium to help in understanding storing and communication of spatial associations and form of geographic phenomenon (Neumann, 2012). The multidisciplinary usage of maps could categorize into two main roles i.e. map as a tool for analysis, problem solving and decision making 'visual thinking'; and maps as tool for communication of concepts among peoples Dobesova and Brus (2012). "Map make transformations of information about location, direction, distance, height or magnitude, density, gradient, shape, composition, pattern, connectivity, contiguity, juxtaposition, hierarchy and spatial association" (Visvalingam, 1990).According to (Kraak & Ormeling, 2010) the representation of information feature of a 'map' is a synonym of models, which help people to understand the structures of the represented phenomenon. Therefore mapping is not only a rendering, but also a helping tool for the reader to understand the represented phenomenon (Dobesova & Brus, 2012).Wikipedia (2013a) defines, map as a pictorial representation of area showing element relationship in the specific area of interest. Moreover,Wikipedia (2013a) explains the origin of word "map", i.e. map word originates from the medieval Latin 'Mappa mundi', where 'mappa' meant 'napkin' or cloth and 'mundi' the world. So, "map" became the reduced term stating to a two dimensional picture of the surface of the world.

3.2 Basic elements of map composition

The basic elements of maps are used to provide critical information to the audience. These elements include title, scale, legend, body of the map, north arrow, production date, projection used, grid lines, and

information about sources. The placements of these elements vary from map to map and also depending on the user choice.
The map title tells us what the maps is all about? The legend tells and show us what the mean of used symbols on map. North arrow or compass rose tell us about the directions. Scale is used to show us actual distance between two palaces.

3.3 Cartography

Cartography or map-making is the study and practice of making representations of the earth upon a flat surface. According to the (ICA, 1995), "Cartography is the discipline dealing with the conception, production, dissemination and study of maps". The meaning of the term 'cartography' has change with the advent of technology primarily since 1960. Earlier, cartography was defined as 'engineering of maps'. The apparent change emerged with the fact when subject cartography has put into the field of communication science and the initiation of computer technology. In the context of communication science; cartography is define as 'the transmission of geospatial information through map' (Kraak & Ormeling, 2010).

The definition of cartography stated by the International Cartographic Association (ICA, 1973); does not cover sufficiently the widespread and varied concern of the subject (Visvalingam, 1989). Considering the advancement of computer technology, and geographic information system in the field of mapping, cartography is defined as 'the information transfer that is cantered about a spatial database which can be considered in itself a multifaceted model for geographic reality. Such a spatial data base then serves as the central core of an entire sequence of cartographic processes, receiving various data inputs and displaying various types of information products' (Guptill & Starr, 1984). The Education Committee and Teachers Group of the British Cartographic Society proposed "cartography is the science and technology of analysing, interpreting and communicating spatial relationships by means of maps". The map is in general, cover various visualizations including maps, 3D model, globes and chart etc., having different functionalities and therefore different concepts of cartography (Visvalingam, 1989, 1990). Taylor and Almond (1991); stated about the definition of cartography as 'the organization, presentation, communication and utilization of geo-information in graphic, digital or tactile form. It can contain all phases from data preparation to end use in the creation of maps and related spatial information products'.

3.4 Thematic cartography

Thematic cartography is a subgroup of cartography where the production of thematic maps occurred. A thematic cartography is a special kind of map to deliver information about a single topic or theme, such as literacy ratio or population density (Rautenbach et al., 2013). The choice of thematic map is very crucial during the presentation of information as there are various type of thematic maps (see Figure 3-1),each for specific phenomenon, representing different types of data and goals; for example choropleth map and proportional symbol map (Kraak & Ormeling, 2010; Stevens, Smith, & Bianchetti, 2012).

The thematic mapping process of a choropleth and proportional symbol map are shown in Figure 3-3., the starting point is the determination of the goal of the map and the final step is the creation of thematic maps (Rautenbach et al., 2013). The main difference between the two maps is the steps 7 and 8, where color is used for choropleth map and symbol are assigned for proportional symbol map. Therefore designing method of choropleth method is easily extended to other type of maps implementation like proportional map with minor changes.

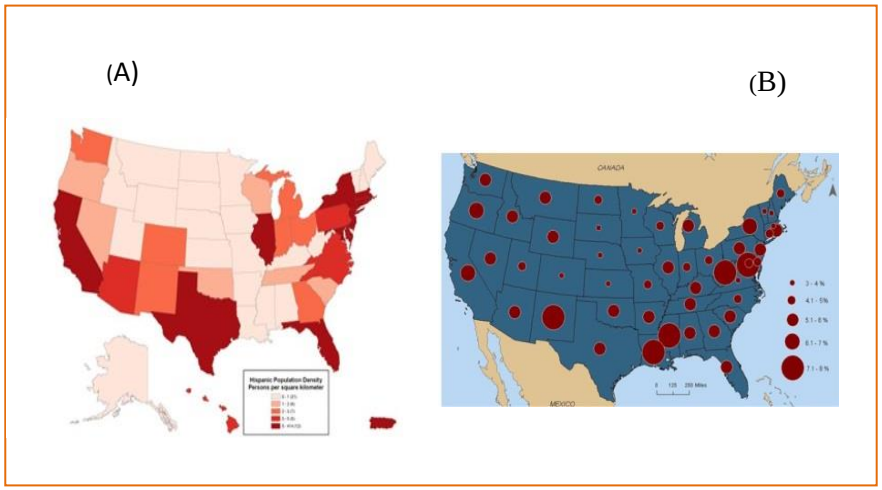


Figure 3-1 (A).Choropleth map, showing Hispanic population density in the U.S. using a single hue sequential colour scheme (B). Proportional symbol map of unemployment percentage in 2000 in the United States (Stevens et al., 2012).

3.5 Steps in map design

Kraak and Ormeling (2010) summarized map design steps, as shown in Figure 3-2. The first steps in this series is analysis of geographic data such considering classification or grouping phenomena, analysing of data characteristics, specifically measurement levels, and determination of absolute or relative quantities in case of quantitative data. The second step is the determination of measurement scale and perceptual properties discussed later in this chapter. The fourth is the selection of visual variable on the basis of measurement scale. Visual variables discussed later in this chapter.

The choice of representation method or type of map is determined after following the above steps.

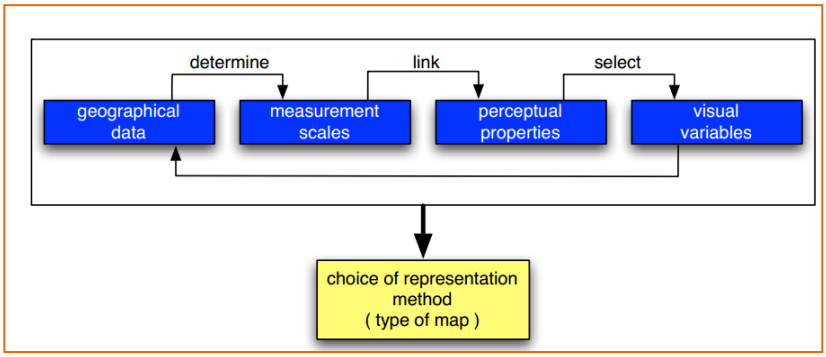


Figure 3-2 Steps in map design (Kraak & Ormeling, 2010)

3.6 Cartographic data analysis

3.6.1 Basics

The determinations of graphics option depend on the nature of information (Kraak & Brown, 2001). Therefore before the construction of map, cartographer should consider nature of data to find out proper Symbology for map visualizations. Maps can answer questions like: Who? Which? Where? When? Why? How many? How much? For these questions to answer we need measured attribute data. Measurement scales play a very important role in science and communication. When we come across to names,

adjectives and numbers all have to define a certain order or properties. For example consider the number "5"; looking to this digit we can think of a certain number of objects, and when we see the word "hot or cool"; certain feeling come in our mind. Therefore, for effective communication these names and numbers should be understood by all the parties who read, use or making the map. Hence, for better communication and correctly addressing these questions we need the careful analysis of data and their represented features (GITTA, 2011; IFGI, 2007).

3.6.2 Measurement Levels

Kraak and Brown (2001), described various steps for analysis of cartographic data. The first step is to find a common denominator for all data which will be used as a title of the map. In the next step measurement scale are used for the analysis of individual components. According to Andrienko and Andrienko (2002) and (Kraak & Ormeling, 2010); measurement scales or attribute values; are usually divided into nominal, ordinal, interval and ratio scale.

- Nominal Scale

According to (Kraak & Brown, 2001), data will be of qualitative or quantitative nature. The qualitative phenomenon attribute are measured on a nominal scale. A nominal scale comprises of only names, which are the consequence of classification. These names are not formally ordered; they define various categories of the equivalent rank. Equating data on a nominal scale comprises only a judgment of equality.

We can say that the data are different in nature or identity, but all information is of the same order. Therefore, we can say that the two data points are equal (=) or not equal (≠). Examples of nominal attributes are (IFGI, 2007; Kraak & Brown, 2001); and this type of representation is called chorochromatic.

- administrative regions (city, county, state, country)
- land use classes (urban, agricultural, forest)
- soil types (podsol, brown soil, laterite)
- difference in gender, languages, religion

- Ordinal Scale

The categorizations of description into formal order are called ordinal scale. This allows us to judge ranking comparison without actual difference in quantity. This lets an assessment of rank about the extent of the variances without providing any information (IFGI, 2007). Ordinal data can be ordered or ranked, but this order or ranking cannot be defined in absolute term. Examples of ordinal attributes:

- opinions on a survey: strongly agree, agree, disagree, strongly disagree
- cool-tepid-hot
- small-medium-large
- unsuitable, suitable, and very suitable
- vegetation height(high, medium, low)

In ordinal scale the numbers are used only for the categorization, we cannot make any mathematical computation with these numbers, such as addition subtraction. For example, a lake with water quality level 2 is not certainly twice as clean as a lake with water quality level 4 (IFGI, 2007).

- Interval Scale

In interval scale both the exact distance and the hierarchy is existing as compare to ordinal scale, where only the hierarchy is available (Kraak & Ormeling, 2010). This means that the result is the arranging of values on scale with approved starting point or zero, and unit of measure. Examples of interval scale are

calendar dates and temperature measured in Degrees, Fahrenheit or Celsius. As these scale are quantitative in nature, therefore some computation are allowed. For example differences can be determined (-) but sums and quotients are not possible (IFGI, 2007).

In summary, interval attribute can be labelling, ranking and difference in amount. No absolute zero point is exists, and defining characteristics is the distance. Moreover, only arithmetic manipulation are +,-.

- Ratio scale

In ratio scale all arithmetic manipulation are allowed. Moreover, the zero point is not a matter of convention, but denotes a natural phenomenon. A ratio scale is the product of quantification through counting, statistics, measurement, etc. The data are represented as absolute values, occasionally containing a unit. Examples of attributes measured on a ratio scale are: population, length, area, velocity.
Generally there are two type of ratio data i.e. absolute and relative ratio data. The absolute ratio data are the outcome of direct measurements or additions of unit i.e. number of family children. Whereas the relative ratio data are absolute ratio data related to other data sets, and they are put in to context, therefore are more informative as compared to absolute data (Kraak & Ormeling, 2010).

3.7 Visual variables

Maps are a form of pictorial communication. After careful selection of measurement scale of data in the previous phase, next we will consider presentation of final map to a broader audience. Therefore we need to find out what kind of symbols, colour will we us? Are labels needed?
Jacques Bertin, a French cartographer and information designer defined comprehensive theories on graphic variables and their usage. He defined six visual variables along with location to distinguish objects in a map (IFGI, 2007). The list comprise of size, value (grey), grain/texture, colour hue, orientation and shape. These graphic variables are the elementary unit of mapmaking. Usage of these variables is also limited having a minimum spread and a maximum length (IFGI, 2007).
Now all the building blocks are available to match the data components to the suitable graphic variable to convey the correct idea. On the basis of measurement scale i.e. nominal, ordinal, interval and ratio; we can pick their corresponding visual variables as are summarized in the Figure 3.4.
The measurement levels are hierarchical, e.g. ratio level as characterized by proportions in Fig: 3.5, implicitly incudes the lower levels also. The figure also presents the size selection rules for geographic objects i.e. points, lines and areas.

3.8 Data classification

According to (Kraak & Ormeling, 2010), classification is the process of categorizations of data on basis of their one or more characteristics. The distinctions of classes are made using various visual variable e.g. using different colour (hue, brightness or saturation).Classification enhances understanding in the data. However, for better human understanding, their number should be limited, as the research has revealed that a maximum of seven classes are easily pick by human at glance (Kraak & Ormeling, 2010). Generally, classes distribution in the range of 4 to 8 is correct and 5-6 is the standard (GITTA, 2011). But the number of classification depend are various factors, like type of symbolization data usage. For example selection of 8 bit scheme is limited to five classes. However, the choice of classified and unclassified map depend on various factors e.g. unclassified is better choice to maintain numerical data relations, where colour shade directly preoperational to the value of each enumeration unit while classified data is the best option for data analysis and accepted by most of the cartographical due to their structured pattern (GITTA, 2011). And also the risk of unclassified data mapping result in unclear visualizations, therefore it is cartographically well come to classified huge amount of data before rendering to user for analysis (Kraak & Ormeling, 2010).

According to Robinson 1976, following are the main reasons of objects or events classification.

- The reductions of large numbers of individuals to small number of groups make illustration and description easy.
- To express phenomena, classes are demarcated about which general declaration can be made.

There are several ways to express classes. However, the method of classifications are generally split into two main categories i.e. graphic approach and mathematical approach (GITTA, 2011; Kraak & Ormeling, 2010).

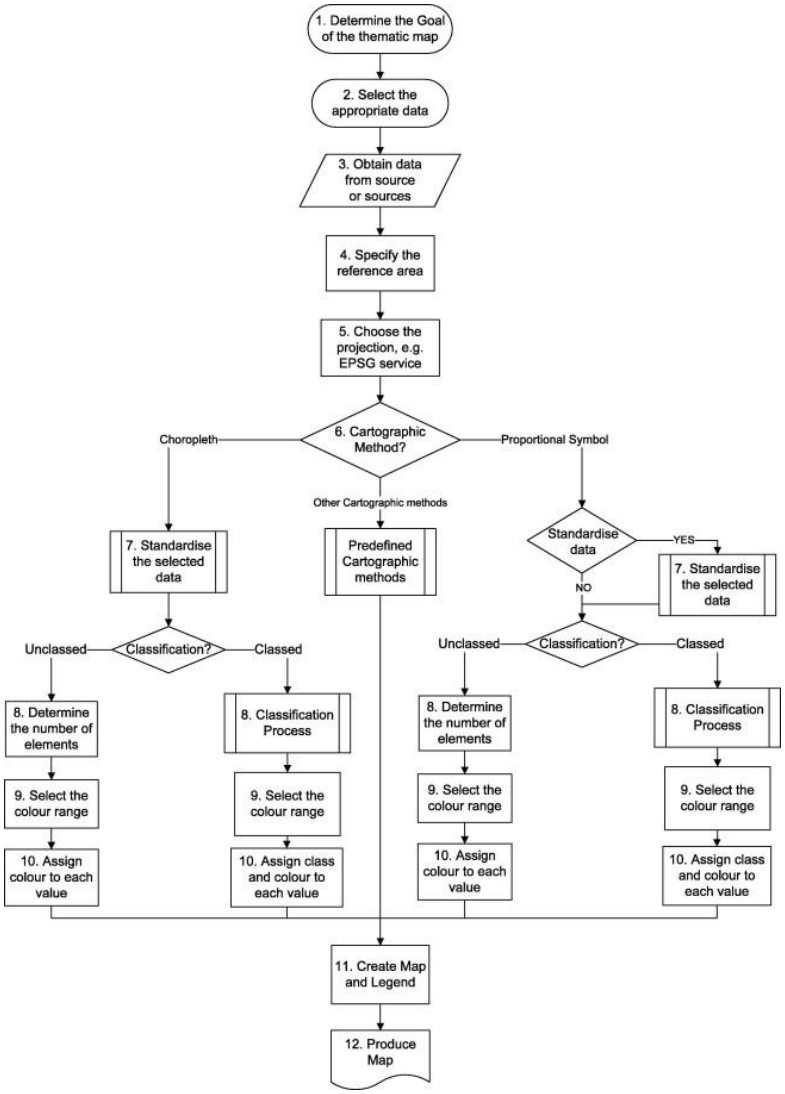


Figure 3-1 Flow diagrams showing thematic mapping process (Rautenbach et al., 2013)]

3.8.1 Mathematical classification approach

In this research work we are using mathematical classification approach named equal steps and harmonic series. These are the most easily computed and one or the other usually the default classification in most GIS software. Equal steps an approach is more adequate to display data that varies linearly, i.e. data with no outliers which cause to skew mean of data far from the median. Harmonic series used with low value observation data (Kraak & Ormeling, 2010).

- Equal Steps approach

In this method the width are equal between the classes. This type of classification especially useful when enumeration areas are nearly equal in size (GITTA, 2011). In this approach, the lowest value is subtracted from the highest value and the result is divided by the number of classes as shown below (Kraak & Ormeling, 2010).

For example the highest value is equal to 8671 and the lowest value is equal to 11, then (8671-11)/5= 1732. The resulted value is used as constant C in the formula below to determine class boundaries.

Formula:

Lowest value + C + C + C+ C+ C= highest value (3.1)

The table in the Figure 3.3 gives the boundary values when the highest and lowest attribute value is 8671 and 11 respectively, using 5 number of classes and corresponding choropleth map cab be seen in Figure 3.4.

- Harmonic Series approach

According to (Kraak & Ormeling, 2010), "a series is defined as harmonic when the reciprocal values of the terms can be defined as an arithmetic series". To define class boundaries from the difference of the reciprocal values of the highest and lowest value and divides result of the two by the number of class as shown below.
For example the highest value is equal to 8671 and the lowest value is equal to 11, then (1/8671-1/11)/5= 0.01815. This results in the series ratio C.

Formula

Reciprocal highest value – C = (Reciprocal highest value – C) – C) = (Reciprocal highest value – C) – C) – C), and so on. (3.2)

Inverse is calculated from the resulting values and used as class boundaries (Kraak & Ormeling, 2010)

Table 3-1 Classes boundary values based on equal interval classification

Equal Interval/Steps Classification	
30772.0	57504.2
57504.2	84236.4
84236.39	110968.59
110968.59	137700.8
137700.8	164433.0

.

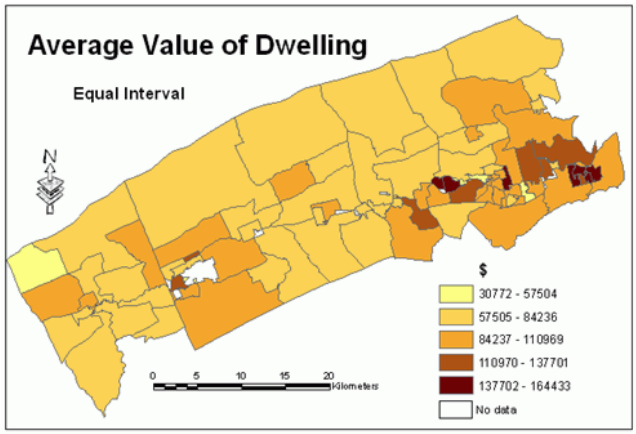


Figure 3-2 Choropleth map based on equal Interval classification (Ela & Dramowicz, 2004)

3.7 Choropleth map

A choropleth map is the most common type of thematic map. "The choropleth mapping technique, which uses "ranges" or "graduated colour," is a type of thematic mapping that focuses usually on a single theme with data summarized by statistical or administrative areas" (Ela & Dramowicz, 2004). The word 'Choropleth' is the combination of two Greek words, choros for 'area' and plethos for 'value'. Therefore, in choropleth mapping area are represented through values (Kraak & Ormeling, 2010). Choropleth mapping is defined by the International Cartographic Association (ICA) as "a method of cartographic representation which employs distinctive colour or shading applied to areas other than those bounded by isolines. These are usually statistical or administrative areas." Choropleth maps portray area through filling the entire area with shade of colours. Usually the data are group into various "class"(range of data values), and for each class different fill is used. The purpose of choropleth maps is to visualize the geographical distribution of the volume of data, and the choice of fill show the magnitudes from high to low by depicting the colour from dark to light respectively as in Figure 3.2 (A) (Stevens et al., 2012).

There are two main types of choropleth i.e. density maps and non-area-related ratio maps. In density type map area are accounted for in the denominator to represented the ratios while in non-area-related map, the area are not accounted as for example the people over 60 in the total production (Kraak & Ormeling, 2010). The construction of these types of maps, required attention as the area is not encountered; which might lead to distorted map quality.

The construction process for both types of maps is shown in Figure 3.6. The initial point will be absolute values such as the number of people or the number of students. To determine the ratio of these absolute values, they are put into perspective such the size of the areas, from where these numbers are collected. In the next step classification of data occurred; and in final round the area of specific category are fill with grey value (Kraak & Ormeling, 2010).

3.9 Concluding remarks

In this chapter we described existing well-structured thematic chorographic rules with special emphasis on the production of choropleth maps. The constructions of cartographically correct choropleth maps need valid implementation; therefore the map maker should have cartographic knowledge. But for the novice mapmaker it is difficult to learn and include cartographic rules in their maps designing. Therefore, to give design relief to novice users, we are trying to put choropleth map design rules in computer system, in such a way that choropleth maps generates with little intervention of users. To give intelligence to computer systems, various approaches are available each has their own pros and cons. One of the approaches is a web service and exposes their operations through interface for consumption. Consequently, in the next chapter we are representing choropleth design rules in computer programming paradigm, to construct a web service and make their usages possible for large communities.

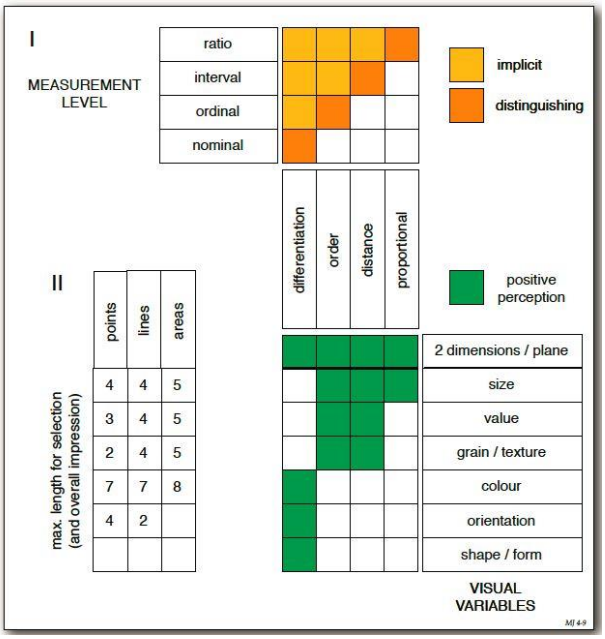


Figure 3-3 Cartographic information analysis (Kraak & Ormeling, 2010)]

4. STRUCTURING THEMATIC CARTOGRAPHIC RULES FOR VISUALIZATION SERVICE

This chapter presents an overview of algorithms for capturing existing choropleth design rules. The aim is to develop a visualization service to create choropleth visualizations according to cartographic rules. Visualizations depend on various factors, such as dataset type, measurements levels, visuals variables and classifications. Therefore, cartographically correct decisions are essential to create accurate mapping. Hence we present algorithms to capture these rules. These algorithms are combined to design visualization service. The service takes decision according to the instructions described in algorithms. The service produce visualization decision as a formatted text, copied to specified location for further usages.
The chapter is organized into four sections. Section 4.1 describes general overview of mapping automation process. Section 4.2 cover workflow for visualization decision, that comprise of four sections; including how to retrieve dataset and determine their type, also discussion on their measurement scale and visual variable to represent that data and similarly the concept of data classifications to make data visualization more meaningful Section 4.3, cover how to use these algorithms to design visualization service and expose their functionality and in last discussion is concluding in Section 4.4.

4.1 NeedsoOf mapping automation process

Nowadays development of digital cartographic tools, and proliferation of open source mapping software are easy accessible for the construction and publishing of digital maps, Dobesova and Brus (2012).Production and publishing of digital mapping is now uncomplicated process. Therefore, it is more significant to transfer cartographical knowledge from cartographers to computers system (Dobesova & Brus, 2011).Because non-cartographer could published maps on Internet, which are very far away from cartographic rules, and mislead the readers (Smith, 2010).However, designing cartographically correct maps require cartographic skills including consideration of data, visual variables and design layout (Kraak & Ormeling, 2011). Therefore, to design maps, mapmakers must have the cartographic knowledge. Moreover, most of the prevailing mapmaking web software application does not provide cartographically rich web maps (Smith, 2010). Hence, transformation of cartographic knowledge becomes more vital, and entire stakeholder can communicate with each other and benefit from shared expertise. But knowledge transformation process is one of the most challenging tasks for object oriented software programmers (Dobesova & Brus, 2011). The design of cartographic service could facilitate in decisions according to cartographic rules, as a step toward visualization as a service (Dobesova & Brus, 2012). In consequence, the use of cartographic rules for the development of semi-automatic cartographic mapping service could benefits widespread users, without comprehensive cartographic and underlying technology requirements.

4.2 Workflow for the creation of visualization decisions

The workflow shown in Figure 4-1 comprise of different algorithms for the design of map. The steps required to design a map were described in section 3.4. However before to design algorithms for different classification of data and visual variables, we need to determine type of spatial data i.e. their geometry. Hence we need connection with dataset to retrieve data as requested. Therefore to produce visualization rules or decision, we divide the workflow into the following main sections.
(a) Data retrieving and geometry type determination; (b) Data measurement scales algorithms; (C) Design of visualization service.

4.2.1 Data retrieving and Geometry type determination

Geospatial data refer to feature objects line, point or polygon; in case of choropleth visualization polygon data type are used, so we need to determine their geometry type prior to process. Consequently we need to include function for this purpose, such as polygon data type is accepted for choropleth visualizations.
A function to determine the nature of data objects is shown in algorithm of Table 4.1 All the steps of this algorithm is described with lines numbers, such as at line 5 *while* condition is used to retrieve data from geometry i.e. 'geom' attribute and assign to string variable at line 6. Line 8-12 find out type of geometry and in case of polygon type it continues process.
The design of this algorithm handles arbitrary dataset. In case of arbitrary dataset we are unaware about the number and type of data attribute, consequently first we retrieve metadata of the dataset and loop to return attributes name, except 'gid' and 'geom' as shown at line 13-25 . These two attributes are excluded, because they have no direct connection with the user. This algorithm returns dataset geometry type and attributes names to the user shown at line 30.

4.2.2 Data measurement scale

The next step in visualization workflow is the determination of measurement scale. With this scale, objects attributes are measured for representation discussed in section 3.4.2.
In this research work we assumed as stated in section 1.2, that the user knows data measurement scale in advance. Therefore we expose this option to the user, also shown in the workflow of Figure 4.1 (b).

4.2.3 Selection of visual variable

Maps are a form of pictorial communication discussed in section 3.5; hence to correctly represent data to wider audience, selection of valid visual variables are vital. In case of visual variable *'colour'* various methods can be used to assigned colour integer. Using choropleth representation we need one shade of colour to represent ratio data discussed in section 3.7. Consequently to assign colour integer using RGB colour scheme, various approach can be used, such as we can change one colour byte at a time and keep the rest fixed. However in RGB colour system, each colour is of 8 bit, changing one colour at one time limits classification number discussed in section 3.6. In second algorithm we can change each colour byte to provide more flexible colour combination i.e. red, green and blue of 8 bit each, which have integer values from 0 to 255; makes 256*256*256=16777216 possible colours. First concept is used in the following algorithm; also in implementation chapter we implement the second approach.

4.2.4 Data classifications algorithms

Classification is the process of categorizations of data on basis of their one or more characteristics as described in section 3.6. Therefore we need algorithms to classify data and represent to user in better way.
Each data measurement level has distinguished characteristics described in section 3.4.2; as a result we need distinct algorithms for each type of data representation. Afterward to invoked correct data classification algorithms the method is discussed in Table 4.2. Data measurement scale is passed as an argument to the method, and tested at the start, in case of matching, loop body execute.
The data measurement scales are divided into nominal, ordinal, interval and ratio scales, as stated in chapter 3.4.2. Each phenomenon attributes have features that led to choice of scale. Likewise data need to classified, as mapping unprocessed data resultant in uncertain visualizations as describe in section 3.6. Next step is to determine data variables discussed in section 3.5. So, we need to design algorithms to classify the data and to determine their visual variable. With respect to this we used three types of algorithms shown in Figure 4.1, for ratio and interval data we design algorithm shown in Table 4.3.

Ratio and interval classification

To classify ratio and interval data discussed in section 3.4.2, we can use various classifications approaches such as equal steps and harmonic series described in section 3.6.1. The algorithm for equal steps is shown

in Table 4.3, and that of harmonic series in Table 4.6, where required variables were declared and prerequisite parameters were passed as discussed in section 3.6.1.

For the visual variable *colour* assignment, we used Red Green Blue (RGB)[11] colour system. In this algorithm we change one colour at time. We assigned integer 20 to colour variable 'cl' at line 6. The classification of data and colour assignment is performed in line 8-17; where we used loop to continue on basis of number of classes shown at line 9. To find the class bounders for arbitrary classes the logic is shown in line 21-13. The assignment of *colour* to these classes is shown at line 15. To format these output classes with colour as needed, such as use comma delimiter or xml type format, is shown at line 16. Line 17 is controlling logic for classes. The final result is returned at line 19.

To use harmonic series we need to determine value of *C,* such as to subtract reciprocal of minimum value from the reciprocal of maximum value of the intended attribute and divide by number of classes, as described in section 3.6.1; algorithm of harmonic series approach is shown in Table 4.6. Inputs of this algorithm are the maximum and minimum data attribute along with choice of classification number, while output are decisions or rules of data visualizations in string form. Each step is shown with line number and description.

Ordinal data classification

Ordinal data can be ordered or ranked, but this order or ranking cannot be defined in absolute terms, as discussed in section 3.4.2

The algorithm used for ranking or ordering ordinal data shown in Table 4.4. Input to algorithm is the distinct records and number of data attribute (column). Distinct records used to order data. The output of this algorithm produces visualization rules. Variable are declared at line 2-6, where variable 'r' is used to hold colour value range i.e. 0-255. Variable 'c' hold number of records in an attribute. The logic of colour assigned to distinct records is shown at Line 6-14. Loop is used to iterate until exceed number of records in column see line 7. Integer number 255 is assigned to variable 'r' at line 8. Inside 'for' loop we used 'while' loop and it continue to iterate up to the availability of records shown at line 10. At line 11 in first iteration one distinct record with colour is assigned to variable 'order' and line 12 the variable 'r' is decremented to required value such 20 or 50. The iteration of *while* is continued until the condition is true as shown in line 10-13. The output of this algorithm is string of rules/style, which can be formatted as needed.

Nominal classification

Nominal data are different in nature or identity, but all information is of the same order; cartographically we used visual variable *'colour'* for their representations as discussed in section 3.4.2. The algorithm for nominal data is shown in Table 4.5. Required variables are declared at line 1-6.The logic of *colour* assignment are shown at line 8-11. The value of *colour* generates randomly each time. This process is continued when the condition is true. The rules are return at line 14, which can be formatted as desired. The nominal data representation is called chorochromatic map where aerial distribution is shown by distinctive colours, as discussed in 3.4.2. However this algorithm has limitation such that visual variable colour is randomly allotted and each time with different variation.

Table 4-1 To retrieve arbitrary dataset attributes from database

Input = [dataset]

Output=[return column geometry type and attributes names except column name 'gid' and 'geom']

1. begin	
2. gt	// result set object to get geometry attribute 'geom' from'spatial

[11] http://en.wikipedia.org/wiki/RGB_color_model

```
    table
3.  begin while loop
4.  rg                                  // declare variable as string to hold geometry name in text
    format
5.  while(gt.next)                      // retrieved data from gt (geom)object
6.     rg = getString(rg)               // convert 'geom' into string format
7.     begin if loop
8.     if (rg != 'polygon')             // geometry type is not equal
9.        "return "Invalid dataset, please upoald valid shapefile with polygon geometry"
10.    end if
11.    else
12.      return rg='polygon'
13.    ds                               // variable declaration to hold dataset from database
14.    ds=(select data from table)      // retrieve data from table
15.    md                               // declaration of variable to hold meta data of the retrieved
    dataset
16.    cc                               // declaration of variable to hold counting of data
    column
17.   md = getMetadata(md)              // meta data assigned to variable 'md'
18.   cc=getcolumncount(md)             // column number are count and assigned to variable
    'cc'
19.   begin for loop
20.      for ( i=1 to upto cc increment i with 1) // for loop with <=column number and 'i' +
21.       cn                           // declaration of variable 'cn' to hold column name
22.      begin if condition
23.       if (cn =='gid') or (cn=='geom')  // return all attribute to user except 'gid' an 'geom'
24.       Continue
25.      end if
26.      ft                            // declare variable to hold string comma delimiter text
27.      ft = cn + "" + cn
28.   end for loop
29.  end while loop
30. return ft
31. end
```

Table 4-2 To invoke method base on measurement scale

Input:[data measurement scale]
Result: [invocation of related method]

```
1.  begin
2.  ms                                  // declare variable to hold measurement scale
32. begin if
3.  if (ms =='ratio' OR ms=='interval')     // check both conditions
4.    call ratioandinterval method          // ratio or interval method invocation
5.    return output                         // return message of method
33.   end
34. begin else-if
6.  else if (ms =='ordinal')
7.    call ordinal method                   // ordinal method invocation
8.    return output                         // return message of method
```

35. ***end***
36. ***begin else-if***
9. else if (ms ==’nominal’)
10. call nominal method // nominal method invocation
11. return // return message of method
37. ***end***
12. ***end***

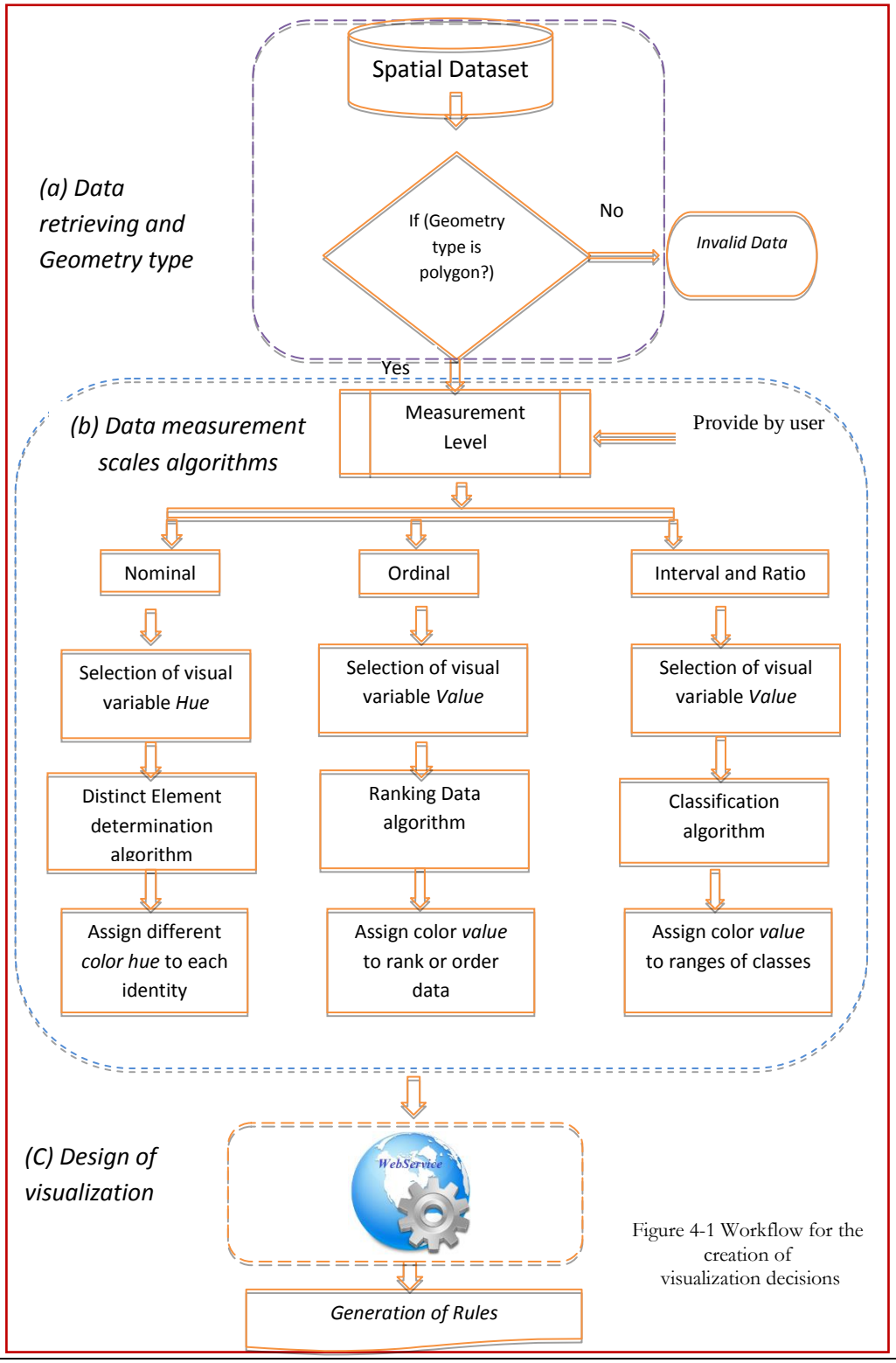


Figure 4-1 Workflow for the creation of visualization decisions

Table 4-3 Classification of ratio and interval data measurement using Equal Steps approach

The formula for equal steps classification:
Lowest value + C + C + C+ C+ C= highest value
To calculate constant C=highest value – lowest value divide on number of classes

Lowest data value as LV // variables declarations

Highest data value as HV

Number of classes as N

Input: [Minimum and Maximum attribute data, Number of classification]

Result: [return classification and colour style rules]

```
1.  begin
2.  LV                          // declare variable LV to hold minimum data quantity
3.  HV                          // declare variable HV to hold maximum data quantity
4.  N                           // declare variable N to hold number of classes
5.  C = (HV-LV/N)               // set variable C to hold constant value for formula
6.  cl=20                       //set variable 'cl' to 20 for colour distribution
7.  Cls                         //declare variable 'Cls' to store return classification
8.    begin loop
9.     for (i=0 to N  increment i)   // loop to calculate data classification
10.    LB                       //declare variable 'LB' to hold low boundary of class
11.    HB                       // declare variable 'HB' to hold high boundary of class
12.    HB = LV + C              // declare both variable to round decimal
13.    LV = HB – C
14.    RC                       // declare variable 'RC' to range of colour
15.    RC = 255 – (C * i )      //set RC to resulted value
16.    Cls += LV + HB + RC      // Assign all calculate values to Cls, we can formatted according to need
17.    LV = HB
18.   end loop
19. return Cls
20. end
```

Table 4-4 Ordinal data categorization

Algorithm:

Input: [Number of data attributes; distinct records from data table]

Result:[return rule/style of ordinal data]

```
1.  begin
2.  r = 0                                   //declare variable to hold colour range
3.  c                                       // declare variable c to hold number of data in attribute
4.  dc                                      // declare variable to hold distinct data attributes
5.  order                                   // declare variable to hold the ordering style
6.  begin for loop
7.    for (i=0 i<= c increment i)           // iterate until number of data count exceed
8.       r = 255                                    // assignment of  colour value
9.     begin while loop
10.       while (dc is not empty)                   // condition to execute until there is no record
11.        order +=dc + r                 // assigned the distinct data with colour to variable and formatted as needed
12.         r -=50                        // decrement the r value , to allot different color to each rank, class or order
13.   end while loop
14.  end for loop
15. return order
16. end
```

Table 4-5 Nominal Data Identification

Input = [dataset name, data attribute name]

Result = [nominal data representation rules]

```
begin
1.  r          // declaration of variable 'r' for red colour holding
2.  g          // declaration of variable 'g' for green colour holding
3.  b          // declaration of variable 'b' for blue colour holding
4.  nd         // declaration of variable 'nd' for nominal data classes
5.  nr         // declaration of variable 'nr' to hold nominal rules
6.  rs         // object to hold retrieve distinct record from dataset
7.  while(rs is not empty)  // iterate through  records
8.    nd=get string( data attribute) // assigned records from column name to 'nd' as string
9.    r=random(255)                  // generate random number for red, green and blue colour
10.   g= random(255)
11.  b = random(255)
12.  nr = nd + r +g+b        // assigned style/rule to variable 'nr' and formatted as needed
13. end while
14. return nr
15. end
```

Table 4-6 Classifications of ratio and interval data measurement using Harmonic Series approach

Formula: reciprocal highest value –C =(reciprocal highest value – C) – C) = ((reciprocal highest value – C) – C)-C), and so on

Input = [maximum and minimum attribute data, number of classification]
Result = [return rules/style of classification]

```
1.  begin
2.  rmax                    // declaration of variable to hold reciprocal of maximum value
3.  rmin                    // declaration of variable to hold reciprocal of minimum value
4.  Cv                      // declaration of variable to hold constant value used for colour value
    distribution
5.  C                       // declaration of variable to hold value of C in formula
6.  Rc                      // declaration of variable to hold colour ranges
7.  Cls                     // declaration of variable to hold classification value
8.  N                       // declaration of variable to hold number of classes
9.  max                     // declaration of variable to hold maximum value
10. min                     // declaration of variable to hold minimum value
11. rmax = 1/max            // calculate reciprocal value of maximum number and assigned to rmax
12. rmin = 1/min            // calculate reciprocal value of minimum number and assigned to rmin
13. C = (rmax –rmin)/ n     // formula C calculation
14. begin loop
15.    for (i=1 upto n  increment 1 to i)
16.    maxb                 // declaration of variable to hold absolute maximum class boundary
17.    minb                 // declaration of variable to hold absolute minimum class boundary
18.    maxb = abs(rmax – C )  // maximums boundary
19.    minb = abs (rmin – C)  // minimum boundary
20.    rc = 255 – (cv * 1 )   // determination of colour ranges
21.   Cls = minb + maxb + rc  // assignment of all rules to variable Cls
22.   rmin - = C
23.  end for loop
24. return Cls
25. end
```

4.3 Design of visualization Service

Visualization as a Service (VaaS), can be defined as a loosely coupled visualization service for the production of on demand visualizations discussed in section 1.2. The term loosely coupled used in the sense that service is independent from dataset, and consequently they can handle arbitrary shapefile dataset. Generally visualization as a service (VAAS) referred to as "on-demand visualizations" i.e. it is based on the concept that the visualizations can be provide on demand to the user regardless of geographic or organizations separation, of data provider and service consumer .

Loosely coupled visualization service design is based on the combination of various methods or services. As discussed previously we can structure already established rules of cartography in various ways. We designed algorithms for different data level (See section 4.2), which can be implement in any programming languages. Also we can design each algorithm as a service and compose in a service. Likewise, we can design one service with various methods for each algorithm. In either way to follow we need to expose service operations for the usage of others applications as Interface. Interface is an entrance point that clients use to access the functionality exposed by the service as discussed in section 2.1.

The conceptual design of visualization service is the combination of required classes with attributes and methods. To design the conceptual model we identity all classes or concepts needed to constructs visualizations service. The algorithms discussed before are used as methods of VaaS. Consequently we illustrate a class diagram to show relationship among classes as shown in Figure 4.2.
The class diagram illustrates the dependency relationship between classes. Dependency is a weaker type of relationships between classes, showing that one class uses attributes of other at some point in time. The independent class consist of parameters or local variables, use by a method in dependent class (Wikipedia, 2014).
In the Figure 4.2, the class consist of name at the top, next are the attributes, and the last part shows operations. The notation (+, -),that precedes the attribute, or operation name, indicates the visibility of the element, in case of + symbol, the attribute, or operation, has a public level of visibility; if a - symbol is used, the attribute, or operation, is private.

4.3.1 Service classes

The service class diagram consists of various classes with methods of specific functionality are describe as follows.

- DataConnection
  DataConnection class intend to established the connection with the database, consist of two methods i.e. setConnection and getConnection. These method are used set the connection passing required arguments, and established the connection.
- DataService
  This class consist of five methods. One is web method i.e. getColumnName, while others are normal programming methods. A web method is expose as a web service operation (ORACLE, 2013).
  Using getHigestValue, getLowstValue methods declare with two parameters to hold two arguments i.e. dataset name and data attribute, and return maximum and minimum data value respectively. In both of method parameter are of type double.
  Geometry type is return when using getGeomType method that need dataset name as an argument. We also need Spatial Reference System Identifier (SRID)[12] value. SRID is a unique value used to unambiguously identify spatial coordinate system definitions. It has integer data type. We also need data attributes name to expose to user, to achieve this task we declare getColumnName web method with one parameter to hold dataset name as an argument.
  In conclusion these class methods can be used in various locations we needed. One the method return attributes column name of requested dataset.
- DataMeasurementLevel
  This method consist of four enumeration value i.e. nominal, ordinal, interval and ratio, which are passed to method when needed will be discuss later.
- VisualStyleService
  VisualStyleService class consist of four methods, in which getStyle is a web method and other three are normal programming class methods. getStyle method declaration consist of six web parameters. The first five are string data type and last one is integer data type. String web parameters are data table name, data attribute name, data measurement type, colour type, label field and integer is used to represent number of classes. This method retrun visualization rules in string format which are formatted are required.
  The other class methods are as under.
  getRatioANDIntevalClassification method is used for the classification of ratio and interval data. This method make used of both classification algorithms described above. The method

[12] http://en.wikipedia.org/wiki/SRID

declaration consists of four parameters. The first two are highest and lowest value of type double, third parameter is the number of classes, and the fourth parameter is the type of colour to choose for visualization. This fourth parameter is used when we make all RGB colour byte variable concurrently for the creation of one shade of colour. In case of one byte variable at time, then there is no need to pass this argument to the method, as shown in Table 4.3. This method returns string of records, which consist of all the decision necessary for the specified visualizations. The string is formatted as needed.

getOrdinalCalssifcation method declaration can of various type. In one type declaration we declare two string parameter i.e. dataset name and data attribute used for classification, and make connection with database to retrieve required records, shown in class diagram.

In other choice we can declare method with two parameter of integer and string type. The integer is used to hold number of records in dataset, and the string is used to hold district records of data attribute (column), as shown in Table 4.4., of ordinal data categorization algorithm. This method return rules of visualization to VisualStyleService.

getNominalClassifcation method also can declare in two ways. In one type of declaration we define two string type of parameter to hold dataset name and attribute name as an argument, also shown in class diagram and in Table 4.5. In other type of declaration we can pass distinct data attribute records only. This method return rules of visualization to VisualStyleService.

- ColorSchem

  This class consist of one method calculateShades with declaration of two parameters. The first parameter is to hold colour object, i.e. red, green or blue as an object, the second parameter hold number of classes. In this method we make all RGB bye variables to make colour choice more flexible. This method returns style as a List of colour object.

- XMLDataStorage

  This method save the visualization formatted in xml form to specified location. This method is called from VisualStyleService for each classification method.

All the above class are used to create a web service to return visualization rules in string format. The rules are needed to use in map file creation. For each above classification method, three methods are used to declare as under.

- RatioANDIntervalMapScript, OrdinalMapScript, NominalMapScript

  This classes consist of one method each i.e. getMapfile with declaration of one parameter of string type and return the creation of map file for each data measurement level. These method first parse xml documents and get stored information and then produce mapfile and save with '.map' extension.

- mapScriptExcute

  This class is used to execute program (explained later)

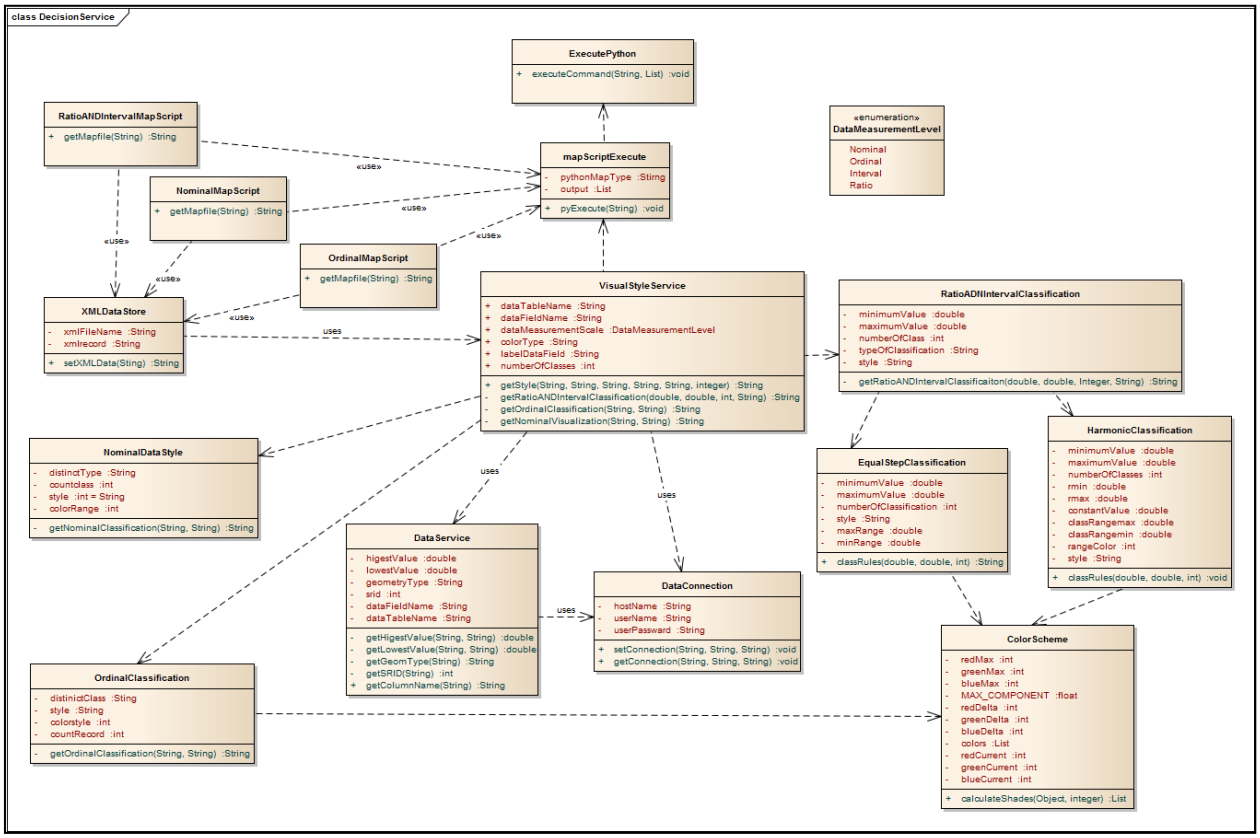


Figure 4-2 Class diagrams of VaaS

4.3.2 Service Interface

Service interface is a contract of specification of behaviour that needs to be fulfilled by the implementer, as discussed in section 2.1. This interface refers to a software service defining the desired map visualizations to the area of arbitrary choropleth mapping.

Visualization service interface is shown in the Figure 4.3; consist of operations to be used by other service or applications. This service interface exposes two operations i.e. getColumnName and getStyle, as explained above. The service operation are summarize in the Table 4.7.

Table 4-7 The operations of service Interface

Operation Name	Description
getColumnName	Return name of attributes of a dataset (table) of the shapefile
getStyle	Return the cartographic rules (style) in text format, based on data measurement scale of the associated layer (data file).

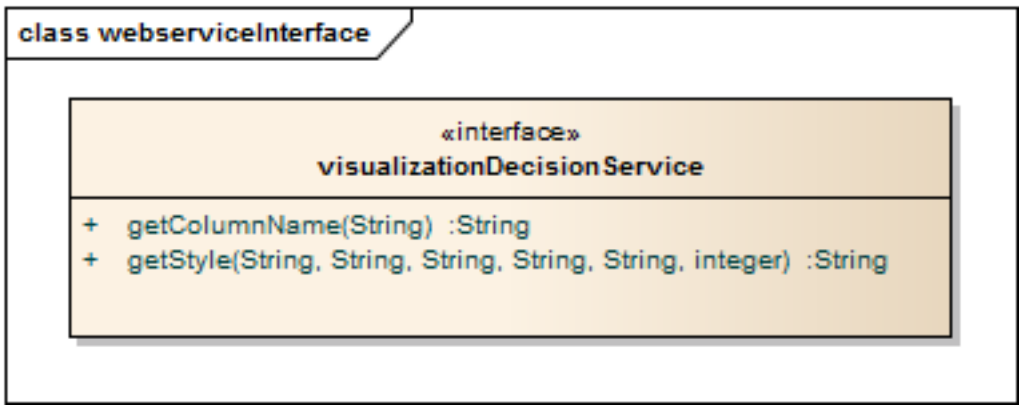


Figure 4-3 The Interface of cartographic decisions Service for choropleth visualizations

4.4 Concluding remarks

A web service is proposed to consider all the decisions required for the choropleth visualizations. The service is composed of three main algorithms for nominal, ordinal and interval/ratio measurement level. For classification equal steps and harmonic series algorithm are used. Class diagram represent the design of the service, the required classes, attributes and operations. Also interface of the service is provided to interact with the service and use their internal functions without knowing complexity of the service. The interface exposes methods which offer attributes name for arbitrary dataset. Also getstyle method return styles/rules for data classification. These information are generated in text format. The output text file holds all the decisions regarding visualizations such as choropleth, chorochromatic. This text file will be used by other program to create map file for the mapserver, to render final visualizations. The design of methods to parse this xml file and generate map file is also shown in class diagram. Next chapter discuss the prosed framework of *VaaS* and the creation of map file in details.

5. DESIGNING A VISUALIZATION SERVICE

In this chapter the framework of the Visualization as a Service-VaaS- is presented. VaaS is a cartographic service that can be defined as a service to process thematic data for various types of visualizations such choropleth, chorochromatic and proportional symbol map. In this research our objective to create choropleth visualization discussed in section 1.2. The main objective of VaaS is to handle arbitrary stored shapefile thematic dataset from a database and visualize according to cartographic rules.
A framework is proposed to include component of visualization service for the creation of visualizations rules and mapfile and render map accordingly using mapserver. We are using open layers as a common graphical Interface for visualizations. The web service development follows SOAP based architecture styling described in section 2.3.1, also the framework using OGC open source specification, Web Mapping Service (WMS), to render final visualization.
This chapter is organized as follows. Section 5.1 illustrate sequence diagram of VaaS to process dataset and deliver the requested visualizations; while Section 5.2 define components of visualization service framework, Section 5.3 conclude the chapter.

5.1 Sequecne diagram of visualization service

The following steps are executed in sequence to generate choropleth visualizations services. These sequences are also valid for other type of visualizations; such chorochromatic and proportional symbol map. The Figure 5.1 illustrates these steps, after establishing connection with the spatial database.

1. The client sends a request for a choropleth data visualizations map to the VaaS, using GUI of the system to dataset from database.
2. VaaS request attribute names of a dataset using Data service class
3. DataService return layer data attributes names in string form to the GUI
4. The client makes selection from dataset attributes names to visualize along with other required parameters such as data measurement type, field to label, colour to display and number of classes.
5. VaaS, VisualStyleService is used to invoke relevant methods, such as RationANDIntervalClassificiton or NominalStyle or OrdinalClassificaiton based on step 4.
6. For data classification 'equal steps' or 'HarmonicClassificaiton' methods is invoked default classification is equal steps.
7. VisualStyleService request to XMLDataStore to save rules as xml document on drive.
8. This xml are parsed through MapScriptService to create mapfile e.g RationANDIntervalMapScript or NominalMapScript or OrdinalMapScript.
9. Map file is read by mapserver to create the requested visualization.
10. Mapserver read data from database according to the expression stored in map file and create choropleth visualization.

5.2 Framework of VaaS

Most of the current applications use a tightly coupled visualizations approach. The tightly coupled methodology offer static visualization from fixed dataset schema, and is confined to defined data schema.

There is lack of a comprehensive visualization service platform to acquire distributed arbitrary dataset, and to visualize classified data to user on demand discussed in section 1.2.
To provide on demand choropleth mapping, we need a flexible platform to accept arbitrary dataset and produce visualizations according to cartographic rules. To handle arbitrary dataset and define their visualizations rules, we need a service that produces visualization rules and that use for the data visualizations.
The framework of VaaS described in Figure 5.2 comprise of two main units. The component to perform decisions and produce rules for visualizations and the next unit use these rules to generate mapfile. These components are fitted within the three layer architecture to produce final visualizations. These layers are data layer, business layer and presentation layer. The two main unit of framework are part of business layer, and described as under.
The decisions component is the web service that composed from various class methods such as DataConnection, DataService, VisualStyleService as discussed in section 4.3 while the other unit of the framework composed from map script class methods as discussed in section 4.3.

The executions of framework component are shown with an example, where the user interested in choropleth visualizations. To get visualizations rules for the interested phenomenon, shapefile are export to spatial database using available tools of GIS.

The VaaS framework connects with database server using *DataConnection* method. User insert dataset name to the system using GUI. The system will use *DataService* methods to return all the column names and expose to user. To achieve this *getColumnName* method of *DataServie* is executed. The user make selection from data attribute to visualizae, along with other required parameters of web method *getStyle* , such as data measurement level, type of colour, label field as discussed in section 4.3. In case of ratio data measurement the *getRatioANDIntervalClassificaiton* method is invoked form *getStyle* method, to process the data using classification method *classRule* of any classification approach class i.e. *EqualStepClassification* or *HarmonicClassifction,* default is equal steps, also during classification process visual variable *'value'* assigned on the basis of generated classes range. In case of colour assignment to used one byte at time then algorithm of Table 4.3 can be used. In addition to that we can use *colorScheme* method to utilized all byte of RGB colour system as shown in Figure 5.2. Classification method return rules in form of string to *getStyle* method, where further information includes to the rules, such SRID value of data layer. Final XML document is structure here, while format of xml generation is shown in Table 5.1. The *getStyle* method invoked *XMLDataStore* class method to stored document on specified location, shown in Table 5.2. The xml document show two classes with boundaries and *colour* integer value. Also name of dataset and other required information are stored with classification style. This document is retrieved using *getMapfile* method of class *RatioANDIntervalMapScript* as shown in Figure 4.2 and described in section 4.3. getMapfile parsed the xml document and get all the stored information to create map file.

To link these two units of framework, and execute process in sequence we make use of *pyExecute* method of class *mapScriptExecute*, from *getSyle* method of class ViauslStyleService as shown in Figure 4.2. *pyExecue* method called the *getMapfile* method to create the required map file, while *pyExecute* method depend on another class named *ExecutePython*, as shown in Figure 4.2.

Map file configures how the map will look, what size it will be, and which layers to show, as discussed in section *2.5*. This file not only consists of style information for the visualization, but also handle client request for WMS to handle arbitrary user requests for visualizations.

Mapfile creations can be reached using various approaches. In one approach we can create map file for each shapefile from xml document locate at specified location. In second approach, we can invoke

methods directly without creating text files and can pass information as a string argument. Both approaches have advantages and disadvantages. The advantage of the first approach is that if for the same resource there are concurrent requests, the server will use stored mapfile without repetition of the whole process to retrieve the data from database and process. The disadvantage is that it does not reflect the change behaviour of dataset. To get dynamic update in data we can use the second approach. In this case whole process of retrieving data from database and creation of mapfile is executed for each request. Therefore combination of both approaches will be more robust.

The productions of mapfile comprised of various steps (See Appendix A). Xml documents shown in Figure Table 5.2 consist of Layer, SRID, Classitem, LabelItem and Colortype names stored as an attribute of the respective element, and geometry type is stored as a child element of layer element. To parse these elements we can use various search methods, such as search element by tag name; where <layer> is called a tag, to get their attribute values. In case of random number of child elements, after searched through tag name, child element value is extracted using loop, such as for loop to explore the whole document. The xml document parsed to produce Mapfile is shown in Table 5.2.
XML parser converts xml document into object. The object is manipulated to extract the informations. Therefore parsing and mapfile creation are shown in separate sections in workflow (See Appendix A).
The workflow consists of two main sections. Xml document are parsed first using available libraries and afterward mapfile are generated. For the creation of map file thexml document stored in object are parsed and searched using xml tag or attributes names. The matched data are stored in declared variables. For example, in Table 5.1, to retrieve attribute value of <Layer> element, first search is made using tag name 'Layer' and accessed attribute or child element data as required. Arbitrary numbers of child elements are search using tags names, and their child element value are retrieved using loop, such as 'for' loop. The 'for loop', search the whole document with specified tag. These search strategy create class boundaries as specified in xml document. The number of classes depends on the data availability in xml document. The extracted classes with other informations are stored in variables. These variables used in MapScript programs as discussed before to produce the mapfile. Each variable is assigned to respective objects, and finally saved in specified location with ".map", extension. The Mapserver read mapfile and act according to the content of the documents to produce the requested visualizations.

Data layer

Data layer comprises of information about data, data format and data source stored in database, using various types of DBMS for instance, PostgreSQL. The data can be point, line or polygon types to represent various phenomena. The service processed shapefile dataset of Esri[13] standard, as discussed in 1.2.

Presentation layer

The presentation layer is simply to access and present the visualisation to the client. It is a graphical user interface to provide bridge between the system and the user. Hence to create GUI for VaaS, we need to create clients to consume the service. We can consume the service in different ways using different technology discussed in section 2.3.
Nevertheless we can make use of OpenLayers API in this layer to overlay thematic visualization as an image using Web Map Service (WMS).

[13] http://www.esri.com/

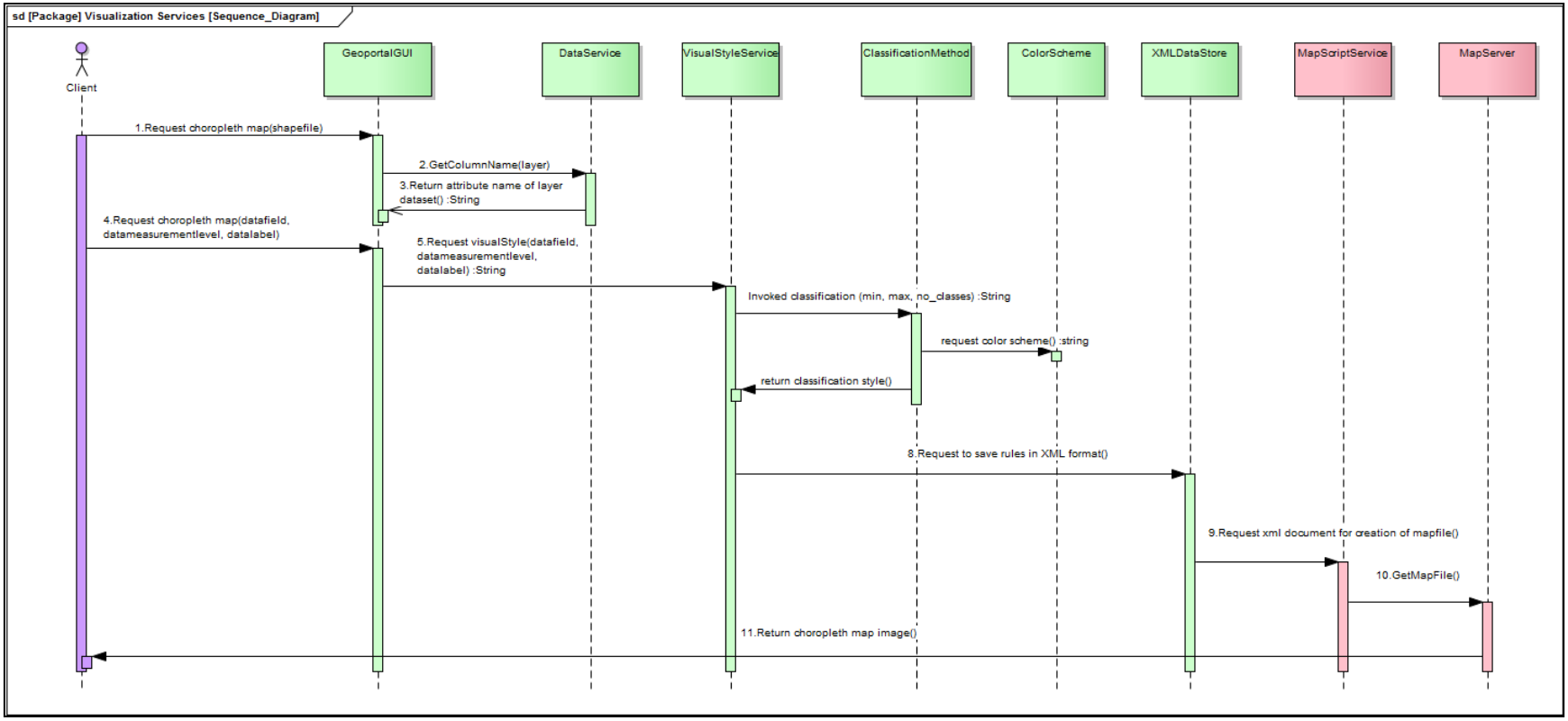


Figure 5-1 Sequence diagram illustrating the choropleth visualization service

Table 5-1 Structure of text file to stored generated decisions of service

```
xml_structure= "\n<Map>\n"
             + " <Layer name=\"" + tablename + "\">\n"
             + " <Type> geomtype </Type>\n</Layer>\n"
             + " <ClassItem name=\"" + dataAttribute + "\">\n</ClassItem>\n"
             + " <LabelItem name=\"" + labeAttribute + "\">\n</LabelItem>\n"
             + " <Colortype name=\"" + colorAttriubte + "\">\n</Colortype>\n"
            + "  <minBoundary>" + upperClassBouundary + "</minBoundary>\n"
            + "  <maxBoundary>" + lowerClassBoundary+ "</maxBoundary>\n"
            + "  <color> " + colorScheme+ "</color>\n</classes>\n";
            + "</Map>\n";
```

Table 5-2 Example of rules for data visualizations

```
<Map>
 <Layer name="worldborder">
        <Type> Polygon </Type>
</Layer>
<ClassItem name="lifeexp05">
</ClassItem>
<NumberofClasses> 2
</ NumberofClasses >
<LabelItem name="name">
</LabelItem>
<Colortype name="Red">
</Colortype>
<SRID name="4240">
</SRID>
<classes Id="1">
        <minBoundary>10.0</minBoundary>
        <maxBoundary>508.0</maxBoundary>
        <color> 235</color>
</classes>
<classes Id="2">
        <minBoundary>508.0</minBoundary>
        <maxBoundary>1006.0</maxBoundary>
        <color> 215</color>
                        …
                        …
                        …
</classes>
</Map>
```

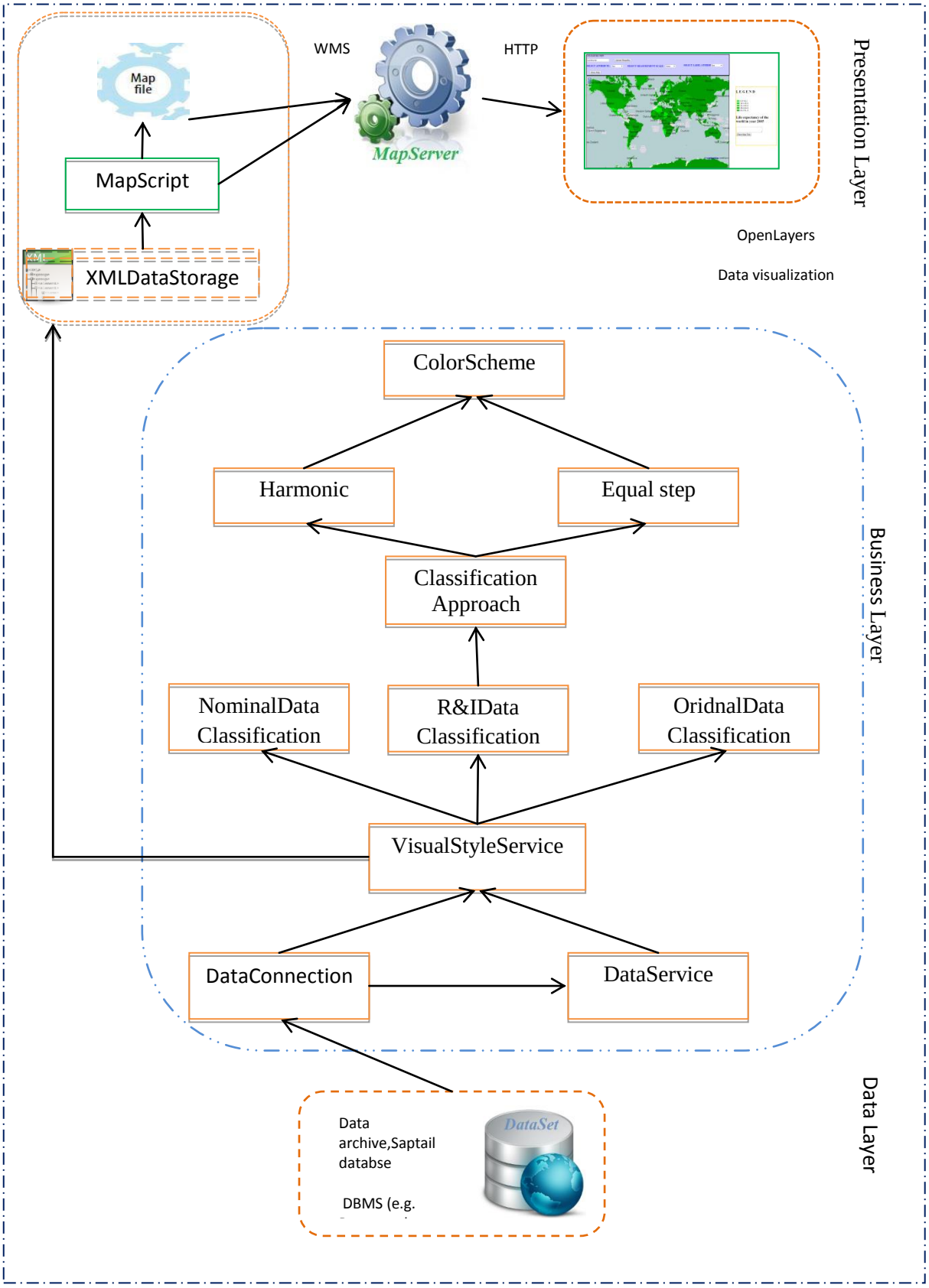


Figure 5-2 Frameworks of VaaS

5.3 Concluding remarks

In this part of the document we discussed the framework of our proposed *VaaS*. The framework is consisting of different types of service methods. This framework can be dividing into two sub sections. One section deal with the production of data visualizations rules, while other part relate to the production of mapfile using that visualization rules. Mapserver read mapfile and act according. The final visualization is overlay in OpenLayers used as base map. To implement and test the VaaS with required technologies are presented in next chapter.

6. SYSTEM IMPLEMENTATION: LOOSELY COUPLED VISUALIZATIONS

The system described in the previous chapter was implemented and tested using different datasets. Details of these dataset implementation described in the following sections. The purpose of our experiments is to investigate the production of loosely coupled choropleth visualizations. The chapter opened with discussion about tools and technologies were used for system development and implementation. In the next section of the chapter, system implementation was discussed, along with source code reference of each method.

6.1 Tools and technology

The tools and technologies used for the development and implementation are shown in Table 6.1. PostgreSQL[14] version 9.2 with extension of PostGIS 2.0[15] was chosen as a DBMS to create and store geospatial database and dumped with shapefile dataset.

For data preparation and storage we used ArcGIS, QGIS Desktop 2.0.1[16] and *'PostGIS 2.0 shapefile and DBF loader exporter'*. PostGIS and QGIS were used to import shapefile into database. Both tools import shapefile into spatial database and stored as a new table of database with name of shapefile. Therefore, meaningful names of shapefile can be useful for automatic map title creation discussed in later sections.

VaaS was developed in Java, using Java APIs for XML Web Services (JAX-WS) a set of Java technologies used to develop Web Services as discussed in section 2.3.1. The concepts of VaaS classes, Interface described in section 4.3, and sequence diagram described in section 5.1 was designed in Enterprise Architect[17] tool.

Python 2.6[18] with mapscirpt API used for the development of various services, such as OrdianlMapScriptService, RatioANDIntevalMapScriptService discussed in section 4.3 and section 5.3; to create arbitrary mapfile base on the contents of xml document. Python 2.6 was used, because Python mapscript was compiled against it and MS4W support this version only. To get information from XML document python minidom API was used; and mapfile was created on the basis of these informations.

Web service was deployment on GlassFish[19] Server, while MS4W MapServer[20] version 3.0.6 was used to published the spatial data and mapping applications discussed in section 2.5, also mapserver support WMS to render final visualization as an image, discussed in section 2.4.

Web Service was consumed using client discussed in section 2.3. For the development of client application and visualization of spatial data we used JavaScript[21] and JSP[22]. The user interacts with the system using

[14] http://www.postgresql.org/
[15] http://postgis.org/
[16] http://www.qgis.org/en/site/
[17] http://www.sparxsystems.com/
[18] http://www.python.org/
[19] https://glassfish.java.net/
[20] http://www.mapserver.org/
[21] http://www.javascriptsource.com/

GUI. GUI provides representation and interaction with the service. Attribute data and resulted visualizations were displayed in GUI.

OpenLayers[23] was used as base map for the final visualizations.

6.2 Implementation of VaaS

VaaS is composed of various methods or services. Each method has specific task to perform; as discussed in sections 4.2& 4.3 and 5.1 & 5.2. Source codes of all methods are shown in Appendix.

Table 6-1 Tools and Technologies used for development and implementation of VaaS

S.no	Tools & Technologeis	Methods/usages	Version	IDE
1	Enterprise Architect	Class Diagram,interface	10	
2	PostgreSQL	Database design,DBMS	9.2	
3	PostGIS	Geospatial Database	2.0	
4	PostGIS shapefile and DBF loader exporter	Loading shapefile into database	2.0	
5	ArcGIS	Geospatial data preparation	10.2	
6	QGIS	Geospatial data preparation	2.0.1	
7	Java EE	Web service implementation	Jdk 7	NetBean IDE 7.4
8	JAX-WS	Java API for XML Web Services development		
9	Python ,Python MapScript	Mapfile development	2.6	pyDev, Eclipse
10	Minidom API	XML document parsing		
11	MS4W	visualizations	6.0.3	
12	WMS	Rendering		
13	GlassFish Server	Web service deployment	4	NetBean IDE 7.4
14	JavaScript	GUI		
15	JSP	GUI		

Dataset attributes are of various types and number, such as string, integer, float, and ten to million records. To retrieve data attributes from arbitrary dataset, we retrieve metadata of the dataset and hence expose column names to user (See Appendix G). In the following snippet of code, metadata was retrieved of the upload dataset, and then ignore the two columns i.e. *geom* and *gid*, from exposing to user.

Table 6-2 Retrieving metadata from arbitrary dataset

```
ResultSetMetaData metadata = results.getMetaData();
int columnCount = metadata.getColumnCount();

for (int i = 1; i <= columnCount; i++) {
   String columnName = metadata.getColumnName(i);

   if (columnName.equalsIgnoreCase("gid")
         || columnName
         .equalsIgnoreCase("geom")) {
      continue;
   }
   finualColumns = finualColumns + "," + columnName;
```

[22] http://www.oracle.com/technetwork/java/javaee/jsp/index.html
[23] http://openlayers.org/

As discussed in sections 4.3 and 5.2 service methods required data from database, such as to find out the highest and lowest value, SRID from arbitrary spatial data. PostgreSQL was used to queries the database and retrieved data (See Appendix G).For choropleth visualizations we need polygon data type. Therefore, we made constraint on data to accept only that data having geometry type polygon, and reject other (See Appendix G.

VaaS consist of different algorithms used in different methods as discuss in section 4.2 and 4.3, and their implementations are given in Appendix G. The output of these methods are shown in Table 6.1, and saved each time as new document.

Structure of visualization rule XML document

The structure of xml documents was designed to fulfil system requirements. The data structure of xml documents are slight different from each other. However each document holds rules of visualizations for different type of data measurement level. As shown in Table 6.3, xml document composed of various information, necessary for requested visualizations. This document was generated from RatioANDIntervalClassificaiton method. The document enclosed all information inside <Map></Map> root element. The root element consist of various child elements i.e. Layer, SRID, LabelItem, ClassItem, Colortype and classes. The layer element has name attribute, to stored name of the shapefile dataset. A meaningful name is encouraged, as it is helped to use in automatic map title creation. Layer element consists of child element <Type>, to hold geometry type of the dataset, retrieved from shapefile database. The srid of dataset is stored in element <SRID>.The correct srid of shapefile dataset is very important for visualizations and also data retrieving. In case of wrong srid, we cannot retrieve or visualize data correctly. VaaS provide arbitrary labelling choice, to get label choice from user was stored in <LabelItem> element. Also to create different classification on the basis of data attribute the element <ClassItem> was used to hold attribute name. Moreover colour choice is hold in element <Colortype>. Data classification enhances understanding in the data as describe in section 3.6. Therefore in xml structure we used element <classes>, consists of sub elements such as <minBoundary> ,<maxBoundary> and <color> in this example; to hold the class boundaries and colour value, allotted on basis of their quantitative data.

The information needed for mapfile creation was stored in xml documents. We used Minidom API[24] to parse xml document and extract the information needed for of designed mapfile (See Appendix-L).

Nevertheless to interact and consume the web service we need create service client as discussed in section 2.3. So, we created a web client with simple Graphical User Interface GUI to visualized data and provide interaction with the service.

6.3 Consuming the VaaS

Web client was designed to consume the Web service. We can create client in different way as was discussed in section 2.3, here we used Net Beans IDE. The creation of web service clients can be synchronously or asynchronously (NetBeans, 2013a), also discussed section 2.3.

6.4 GUI components

Graphical User Interface (GUI) allows users to interact with the system. The GUI, of VaaS is shown in Figure 6-1. The GUI was created to interact and test our system. It shows various controls, such as drops dropdown boxes for selections of attribute values, measurement level, colour, label. The main components of VaaS GUI are:

- Uploading/Selection of dataset

[24] https://wiki.python.org/moin/MiniDom

This control button is used to retrieve dataset fields from the database when user insert name of the shapefile into text box. The fields are list in drop-drown

- Selection Attribute: This drop-drown list all the attributes where the user make selection;
- Measurement Level: This drop-drown list data measurement level where the user make selection;
- Select Colour: This drop-drown list RGB colour where the user make selection;
- Select Label: This control is used to for labelling and users can select field to label;
- Show Map: This control is used to visualize the final map overlaid on OpenLayers;
- Legend: Legend display all the classification with ranges;
- Legend Title: This is a sub title of map, and generates automatically, constitute of dataset name in first line, afterward the field use in classification and the word 'visualization' and
- Enter Map Title: This is use to create customize title for the map

6.5 Loosely coupled visulaiations for Ratio nnd Interval data measurment level

Loosely coupled visualization were implement for different types of data measurement levels. Each data measurement level used their own algorithm, discussed in chapter 4. The ratio and interval classification algorithms (See Table 4.3) were implemented with Java class method named RatioIntervalClassificationService (See Appendix-F & G).

The design algorithm could easy implement for the visualizations of choropleth map. Figure 6.1 (in green colour) and also Appendix-B (in red) show the visualization of 'World Life expectancy for the year 2005', was generated with this method. This visualization consists of five classes with label showing ranges of class boundaries. This method can used for arbitrary dataset, to generate classification of ratio or interval data measurement level. Other visualizations example form different dataset shown in Appendix-C. The map visualized population density of Pakistani province. The colour choice was green, which validate flexibility of colour choice selection. Figure 6-2 shows other type of choropleth visualization i.e. Primary education completion ratio, with labelling. VaaS also provide choice of classification as shown in Figure 6-5, the map shown population densities using equal steps approach with eight classes.

Table 6-3 XML Document Structure with visualization rules

```
<Map>
<Layer name="world_population_density_2011">
        <Type> Polygon </Type>
</Layer>
<ClassItem name="population">
</ClassItem>
<LabelItem name="name">
</LabelItem>
<Colortype name="Red">
</Colortype>
<SRID name="4326">
</SRID>
<classes Id="1">
        <minBoundary>0.0</minBoundary>
        <maxBoundary>234.8</maxBoundary>
        <color> 235</color>
</classes>
…
<classes Id="5">
        <minBoundary>939.2</minBoundary>
        <maxBoundary>1174.0</maxBoundary>
        <color> 155</color>
</classes>
</Map>
```

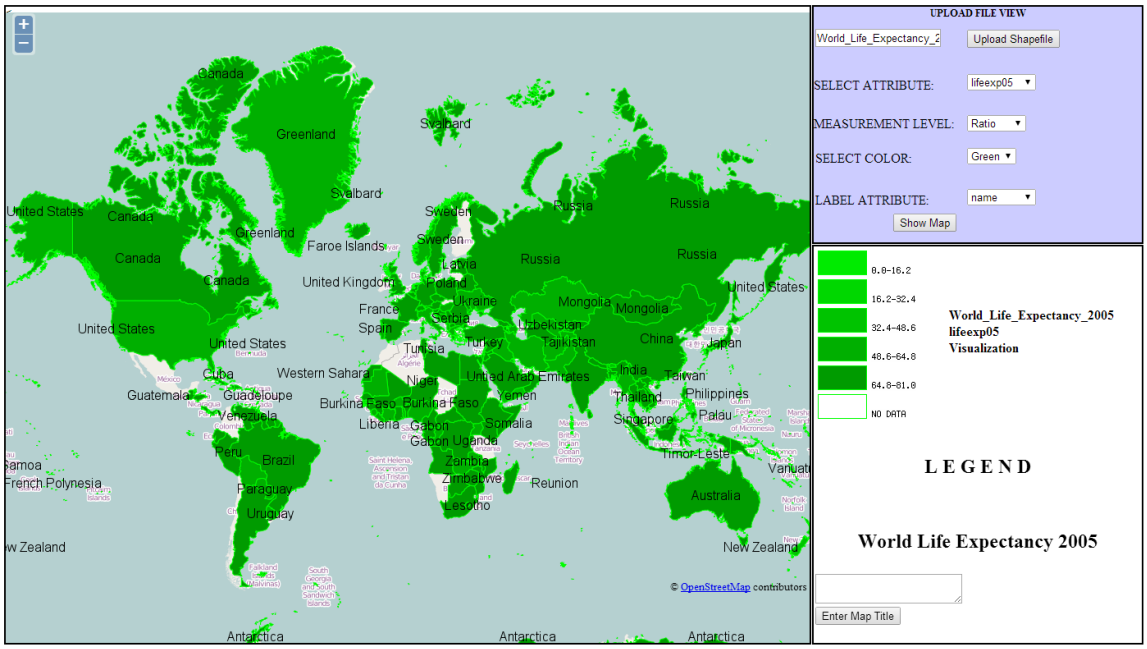


Figure 6-1 GUI of VaaS

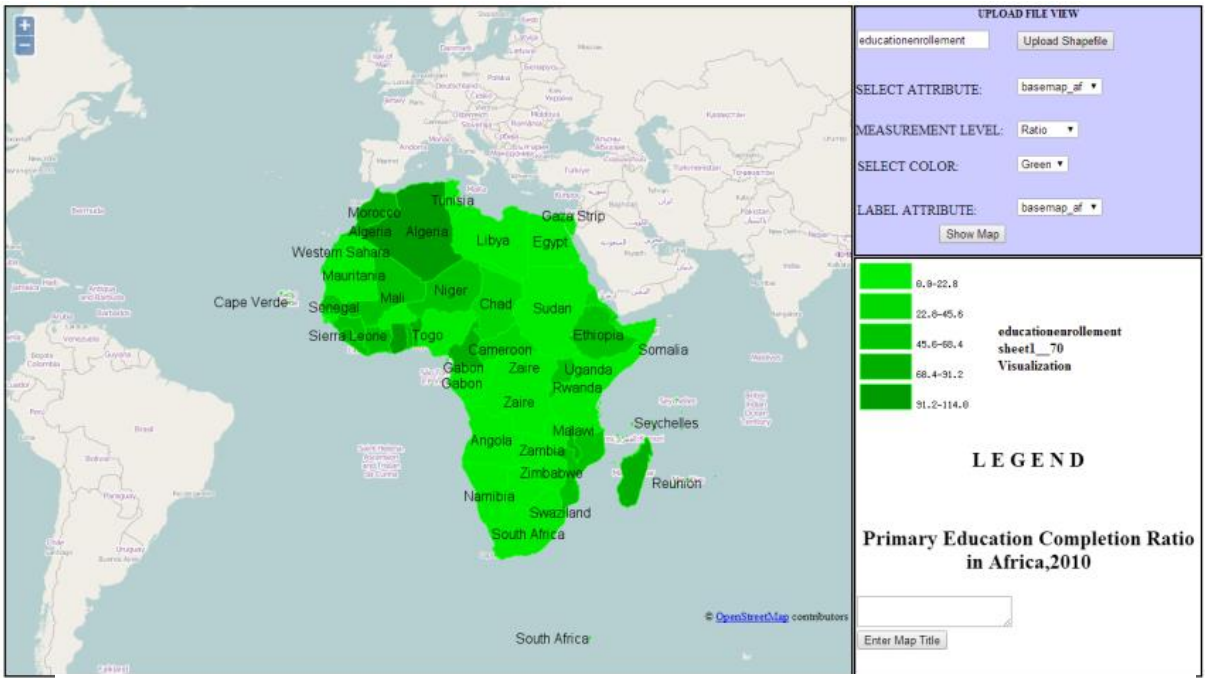


Figure 6-2 Choropleth visualization for ratio data with green colour and label, using RatioANDIntervalClassification method

6.6 Loosely coupled visulaiations for ordinal data measurment level

The data ordinal scale was discussed in section 3.4.2 and their algorithm was described in Table 4.4. The source code for the implementation is shown in Appendix-I. The algorithm was implemented and test on dummy ordinal data. The data comprise of vegetation height at different locations. Numbers were allotted to order data, such as '1' for low, '3' for high, and '2' for medium height. The visualization is shown in Figure 6.3. This method was tested for arbitrary dataset to generate choropleth visualization from arbitrary spatial dataset.

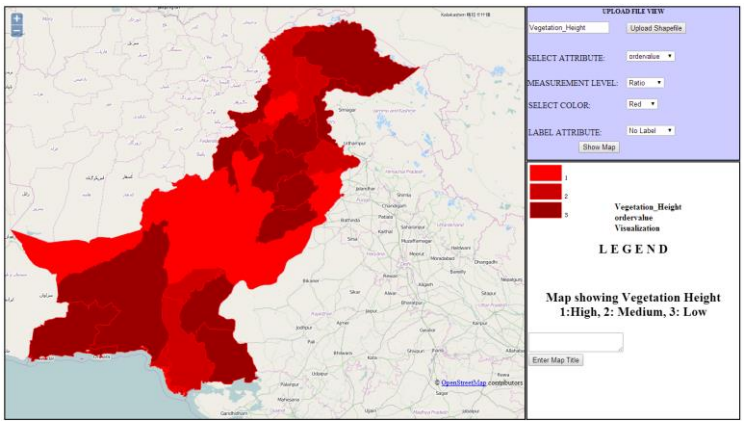


Figure 6-3 Ordinal data visualizations using OrdinalClassification

6.7 Loosely coupled visulaiations for nominal data measurment level

The main objective of VaaS to provide choropleth visualizations from arbitrary dataset, as discussed in section 1.2. However, we can also visualize nominal data. Nominal data algorithm was discussed in Table 4.5 and their source was shown in Appendix-H. The result of nominal visualization method shown in Figure 6.4, a chorochromatic map showed districts administrative boundaries of KPK province of Pakistan. Other examples of visualization shown in appendix-D&E.However in this algorithm random colour scheme was generated from RGB and assigned to each different identity, we have no control on the type of colour.

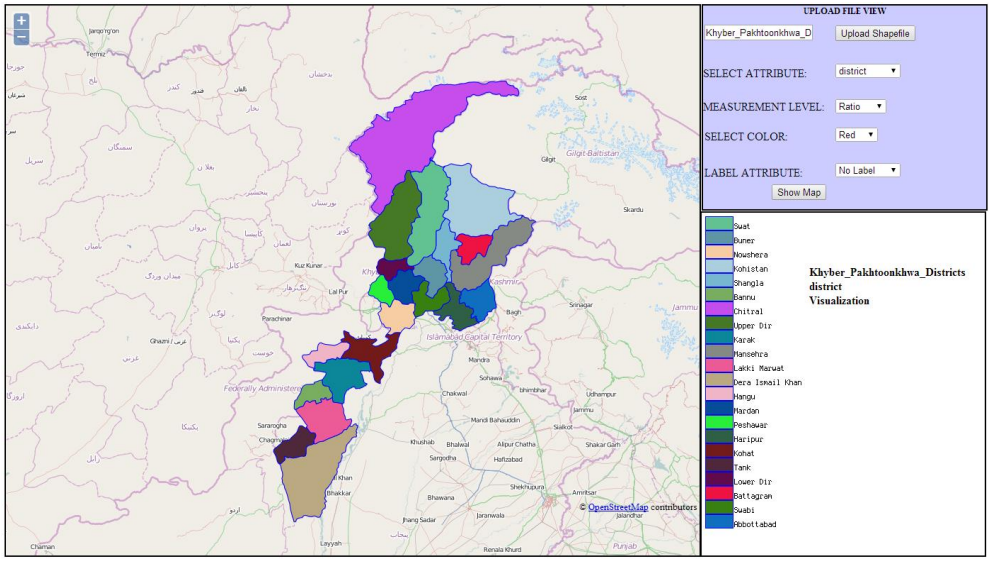


Figure 6-4 NominalData visualization (Chorochromatic map), using NominalClassificaiton method

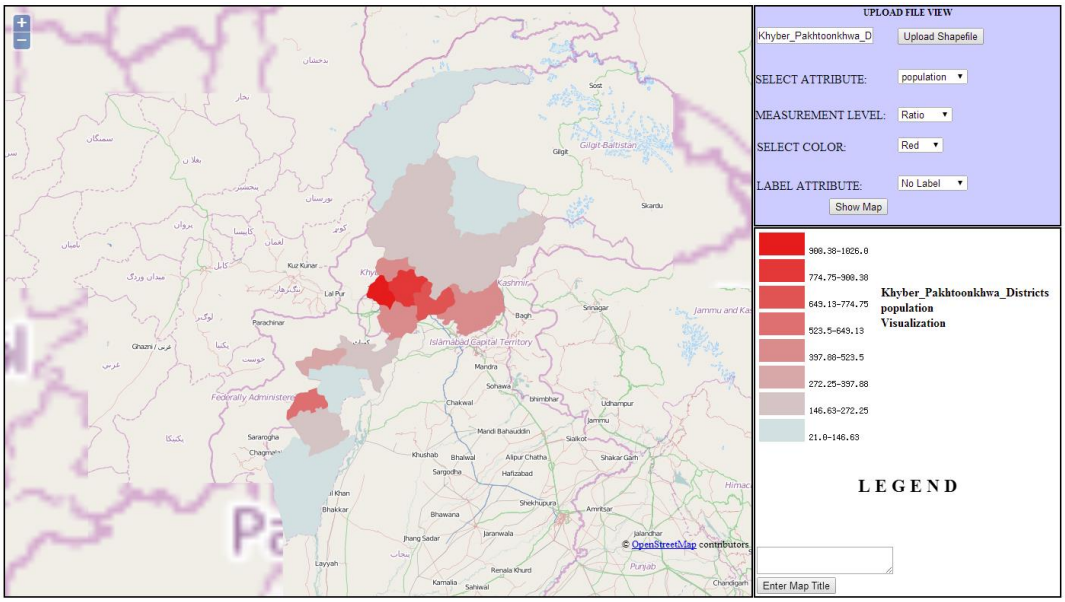


Figure 6-5 Population densities using equal steps approach with eight classes

7. DISCUSSION, CONCLUSIONS AND RECOMMENDATIONS

In this chapter we discuss loosely coupled behaviour of the system in context of arbitrary spatial dataset, along with other relevant points. We conclude over the objectives of this research and give some guidelines for future research work.

In our experiments we used various algorithms to design Visualization as a Service (VaaS), for the production of choropleth visualizations. The design of VaaS was built with composition of various methods based on Simple Object Access Protocol (SOAP) architecture style. The service produced visualization rules in text file that was structure in xml format. These rules were extracted from xml document using python minidom xml parsing library and assigned to respective objects in python mapscript API. Map file was produced and used for visualizations, when transferred to mapserver. The Open Geospatial Consortium (OGC) standard for Web Map Service (WMS) specification was used to overlay the final visualizations on OpenLayers. Discussion on each part outline below in different sections.

7.1 Loosely coupled data association

Traditionally, most of the visualization applications have been design within the boundaries of enterprises i.e. maps are produce for those organization specific applications. The database schema are designed in advance, to stored and map only that defined type of dataset. The applicability of the applications is limited only to that dataset.

The objective is to develop a system that can handle arbitrary datasets for choropleth visualizations. The system is independent from the dataset. To achieve our objective, we designed algorithms to performed specific tasks such as to retrieve arbitrary data set (See Table 4-1), we used PostgreSQL function for this purpose. We encapsulate this function with other methods for the design of visualizations service (See Figure 4-2).The functionalities of the service was exposed to user/developers through service interface (See Figure 4-3); consumption of interface generated arbitrary rules/style for each data measurement level (See Table 5.2); based on ratio and interval data classification. These decisions were used by MapServer program to generate mapfile. The python Mapscript API programs were called from Java to retrieve and generate Mapfile. In Appendix-L the method was developed in python Mapscript API to generated Mapfile. Alike methods were developed for other data level. The generated file was parsed and complied as Mapfile. The final visualizations were produce and overlaid on OpenLayers using WMS with MapServer (See Figure 6.1-6.5).

7.2 Discussion on the ratio and interval data representation using loosely coupled concept

The risk of unprocessed data mapping results in unclear visualization is quite high. The convenient arrangement of data before displaying is cartographically welcome. So, classification enhances understanding in the data. However, for better human understanding, their number should be limited, as the research has revealed that a maximum of seven classes are easily pick by human at glance. For the representation of ratio and interval data measurement level we used two classification approached. Table 4.3 provides algorithm for data classification using equal steps classifications and Table 4.6 was constructed for harmonic series algorithm. The implementation codes of both algorithms are shown in Appendix-C & G. The results algorithms are shown in Figure 6-1 and 6-2. The productions of various

types of maps using arbitrary dataset of ratio and interval data show that they are loosely associated with the system. Moreover, number of classification are customizable.

7.3 Discussion on the ordinal data representation using loosely coupled concept

Ordinal data discussed in section 3.6.For the representation of this type of data we designed algorithms shown in Table 4.4. For implementation source codes see (Appendix-I). The source code shows that we used PostgreSQL to queries the database for distinct records, and also query to find out the number of records. The implementation result of data visualization is shown in Figure 6-3. The algorithm can be used for arbitrary data visualizations. However there are some limitations at this stage. For ordinal data representation we used numeric attribute to rank data. These numeric values are display as label of classes, shown in legend. The colours are visualized with order of number i.e. for 1 bright colour and for 3 dark colours See Figure 6-3. In this case we enter manually the meaning of number such as for High=1, Medium=2, and Low=3. This limitation can be solved in various ways, such to provide other attribute, for classes labelling.

7.4 Discussion on the nominal data representation using loosely coupled concept

Chorochromatic map are used for the visualizations of nominal data discussed in section 3.6. The aim of study was to produce arbitrary choropleth visualizations. However the system also managed nominal data and their visualization e.g. chorochromatic map. The algorithm used for nominal data representation shown in Table 4.5. For implementation of this algorithm see (Appendix-H). The implementations result is shown in Figure 6.4 and appendix D&E, with chorochromatic map. For the implementation of this approach we used PostgreSQL query to retrieve district records and allotted different colour to each distinct identity. However there is one limitation for the allotment of colour. Random numbers of colour scheme were used. Each time data represent with different colours. However this can be solved using the colour scheme implementation in appendix-J.

7.5 Discussion on the concept of color scheme ,label, legned, scalebar and title generation of VaaS

Maps are a form of pictorial communication describe in section 3.1. Visual variables were discussed in section 3.5. In this study we make use of two visual variables i.e. value and colour hue. For the representation of choropleth map we used value, and in case of nominal data representation we used colour hue i.e. chorochromatic map. We developed program for the creation of arbitrary number of colour shade (See Appendix-H).

Visual variable colour scheme was designed more flexible. All byte of RGB were used concurrently to get flexibility in colour allotment (See appendix-J). This algorithm was used in ratio, interval and ordinal data representation. For the representation of nominal data we used another choice algorithm, shown in Table 4.5, and source code of implementation in appendix-H. In both algorithms number of colour shade/hue generation depend on the number of classification or number of distinct records respectively.

A label is text identification of feature on map to help in map interpretation. Label a basic element of map discussed in section 3.1. To include this feature in our system, we expose attributes to user. The user can used system GUI for the choice of labelling attribute i.e. 'LABEL ATTRIBUTE' (See Figure 6.2). The choice of labelling is arbitrary and selection can make from label attribute option in GUI. Mapscript API Label object were used to create labelling. Source code of map file creation using python mapscript API is shown in Appendix-L. However there are some limitations also. The system work in CGI mode, to create mapfile, where mapfile saved with '.map', extension for visualizations. All the information including label

attribute are stored there and mapserver retrieve and visualized these information. Therefore each time the user change choice of labelling new mapfile generate to insert that option in the mapfile. While using API mode of mapserver there is no need of '.map' file storage on server directory, and we render label loosely coupled from other visualizations. Nevertheless, WMS responsible for rendering of map, working in CGI mode i.e. WMS requests are handled by the mapserv CGI program[25]. Therefore CGI mode was used here.

Map legend tells the story of used symbols on map. Legend can created in two ways using mapserver i.e. using GIF legend or HTML legend[26]. Mapscirpt API object 'legendObj' was used to create legend. WMS specifications, offer GetLegendGrpahic operation. The GetLegnedGraphic return a legend image for the requested layer, with label. In case of requested group-name, all included layers will be returned in the legend-icon. But care must be taken while hard-code width and height using legend, as the size of is not known prior to creation.

WMS specification does not provide support for scalebar, similar to GetLegendGrpahic. To include scalebar a functions is need our wms-client.

To generate automatic title we display shapefile name with classification attribute (See Figure 6-5). In this example of visualizaiton, name of spatial dataset located in database with name "Khyber_Pakhtoonkhwa_District" and the name of data attribute "population". We import both these name to system GUI, to add automatic meaning to the legend as whole to the map. These names are change with attribute or dataset. The word 'visualization' is static. So, a meaningful name of shapefile helps in automatic generation of title. We also text filed for customize title writing.

7.6 Discussion on used technologies

Table 6.1 provides a list of used tools and technologies. Different tools and technologies were used at different level. Broadly we can divide these technologies into three levels; i.e. backend, middleware and front end.

The backend level comprise where we created and maintain our database and database server. We configured PostgreSQL 9.2 database server locally, with PostGIS 2.0 spatial database support.

Spatial database was designed on the server. In start of our research work we made assumption about the availability of shapefile dataset within the database. Hence, we prepared and process dataset for various applications. By process of dataset we mean the addition of columns or data into shapefile debase file. To process the data we used ArcGIS and QGIS. The processed data was import into data with the help of 'PostGIS shapefile and DBF loader exporter' and QGIS. Special attention is needed to check the SRID value of the shapefile. We need to insert valid SRID value before import into database, because without this we cannot retrieve and visualize our data correctly. Both tools saved shapefile as a new table in database. By default table are stored with shapefile name. Therefore, a valid name of shapefile, help in automatic generation of map title (See Figure 6-5). Various PostgreSQL queries were used to retrieve information from database (See Appendix).

The middleware is the core part of the system, where all the business logic resides. This layer divides into two main categories, i.e. visualization web service and mapscript API service.

Web service was designed, implemented and deployed using Java technology. The VaaS class conceptual design diagram, shown in Figure 4.2. Java API for XML Web service (JAX-WS), was used for the creation

[25] http://mapserver.org/ogc/wms_server.html

[26] http://mapserver.org/output/html_legend.html#html-legend

of web service discussed in section 2.3. Different algorithms (See Table 4.1-4.6) were implemented in Java (See Appendix). The web serve was deployed on GlassFish Server. GlassFish Server an open source edition provides a server for the development and deployment of Java platform, Enterprise Edition applications and web technologies based on Java technology. The output of web service decision was stored in xml formatted text.

Python mapscript was used for the creation of mapfile. To used python mapscript , we need python version 2.6. Because described in MS4W documentations the support is available for python 2.6 and will not work with earlier or later python versions. We used pyDev, with Eclipse to install python and used mapsciprt. Different mapscript pogroms were designed for each type of data level(Figure 4-2). Also python mindom API version 2.6 was used to parse the xml document.

The methods developed in python mapscript performed two main tasks. The first task in creation of content of mapfile objects using mapscirpt API. Database connection and expression was designed along with colour style. The control structure of colour choice was defined (See Appendix-L). The second task was the design of WMS requests. The requests were create to tackle arbitrary requests from users (also in appendix-L). Both task was saved with dot map extension using the name of shapefile dataset.

Mapfile was used by Mapserver, an open source platform for publishing spatial data and interactive mapping applications to the web. We used MS4W version 2.6 with WMS 1.1.1.

Front end of system comprise of GUI, the container for actual visualizations. To make use of the web service we need a client to make use of the web services operations. We used Java NetBean IDE 7.4. for the creation of client. To consume the web services we used three clients i.e. a Java class in java, a servlet and JSP page in a web application (See Appendix xxx and Appendix). The designed GUI was shown in Figure 6.2. On the right side of legends in GUI we import shapefile name in the first line, and in second line we put the data attribute name, used for classifications. The third line is a fixed word of 'visualization'.

7.7 Discussion on mapfile generation and useage

Mapfile is the core part of Mapserver, which define relationships between objects, and direct the MapServer about location of data, and how to visualize that data discussed in section 2.5. Our objective was to generate mapfile on the fly for arbitrary datasets.

For the production of mapfile python mapscript was used. To achieve our objective we design and implement python mapscript programs for each data measurement level. The classes used for each method shown in Figure 4.2.The implementation of each data level method shown in (Appendix-G-L). The steps used to create mapfile shown in Appendix-F. XML document information was utilized to create mapfile; the output of visualization service discussed in chapter 4.

Two approaches can be used to create mapfile. In one case we can pass generated decisions as strings directly to mapscript methods, without creating mapfile and save on disk. In second approach we produced decisions as text and kept in xml like structure file stored on specified location. Mapscript developed methods retrieve file and parsed to build mapfile, with ".map" extension, and stored. The mapserver read mapfile process and act according.

Both methods have advantages and disadvantages. In first case we can use same mapfile for multiple requests, without repetition of all data retrieving and processing steps. The disadvantage of this approach is that, mapfile cannot reflect dynamic changes of visualizations phenomenon. Considering second approach reflections of dynamic changes propagated into visualizations, on the cost of repetition of all steps for each request. Nevertheless, there are limitations on implementation of both these approaches.

We used two different technologies for the design and implementation of visualization service and mapfile creation; i.e. visualization service took decisions on uploaded dataset, and mapscript program stored those decisions in mapserver readable format called mapfile. We stated that generated decisions were stored in xml format (see Table 5.2), on drive. There are some technical limitations, to pass Java generated decisions text into mapfile generated python mapscript on the fly. As the decisions text was stored with xml formatted structure, therefore passing xml document or string into python as argument creates complexity. Because using utility class for running native executable files, put constraints on some forbidden text (See Appendix-K-1 and K2). Xml file consist of some forbidden text, therefore we cannot pass this type of text directly from Java to python as argument.

There are several options available to pass xml document Java to python. The first option is to use same technology for both types of programs. The MS4W include pre-built support files for csharp, Java, and python mapscript. However, unfortunately python mapscript [27] module contains some classes which are not yet matured in other mapscript supporting languages. However mapscript was compiled and tested against python 2.6, an old version of python series. On contrary Java is more advance technologies for web service creation as compare to python 2.6. In the start of research work we used Java for both cases, but unfortunately during compilation of python program, run time errors were generated. The second approach to pass xml document from Java into python was the used of intermediate software tool and library, such as jython[28]. In this study work we used Java EE with python 2.6 mapscript.

7.8 Discussion on users and uses of VaaS

VaaS a visualization service with an aim to produce loosely coupled visualizations, with the production of interface for interaction and utilization of system methods.

Broadly we can divide the user of the system into two categories i.e. general/application user and application developer. The general users are those users who used the system as an application of visualizations. These users are interested in application side to load shapefile and get visualizations. The second category of users is application developer, who used the service in their own applications. We provide service interface (See Figure 4.3) for the consumption of service. The developer can used any type of technology benefits from the offered services, without the need of understanding of inner complexity of the system.

7.9 Conclusion and recommendations for future research

To achieve the predefined objectives define in chapter 1 several questions have been addressed. The research objectives and questions are reviewed in Section 7.9.1. In the final section, suggestions for further related works are presented.

7.9.1 Finding to the research questions

This research was accomplished by tackling the objectives associated by the following research questions and some of the findings include;

- What is the definition of VaaS within the context of this research?

[27] http://mapserver.org/mapscript/python.html
[28] http://www.jython.org/

Visualization as a Service (VaaS) can be defined as a service that provide loosely coupled visualizations (i.e. choropleth), from arbitrary spatial datasets (i.e. shapefile), stored in spatial database, and to expose their operations to the users through service interface.

Generally VaaS referred to as "on demand visualizations", based on the concept that visualizations can be provided on demand to the user regardless of geographic or organizations separation of data provider and service consumer.

The service generates semi-automatic choropleth visualizations from arbitrary dataset, and exposes their operations through service API, to the user for the consumption of service functionality. This service can be combined with other web service applications.

- Which techniques or specification to use as component of the visualization system?

Equal steps and harmonic series techniques were used for quantitative data classifications. Separate algorithms were designed for both approaches which invoked from respective methods. Both methods return classifications rules or style in text form.

WMS specification was used as one component of VaaS, to render final visualization in image form. WMS support various rendering format such as GIF, JPEG, PNG.

- How to associate data service with the database to retrieve arbitrary dataset?

To established connection with database and to retrieve arbitrary dataset, we create a function that retrieves required arbitrary data attributes or records from database. In order to achieve this first we retrieved metadata of the stored dataset and then expose attributes names to users.

- Which dataset to use for visualization?

VaaS support shapefile dataset stored in spatial database, along with SRID. The shapefiles must according to Esri standard.

- How to formulate rules for different data measurement level?

Algorithms were developed to capture existing visualizations rules for choropleth visualizations. For classifications of ratio data algorithms were designed using equal steps and harmonic series approaches. These algorithms generate data classifications along with colour representation, on the basis of data quantity. RGB colour scheme was used to assigned colour concepts. Two approaches were used to implement colour. In one case one byte of RGB colour change at a time, in second approach all the three colour values were made variable concurrently. In case of nominal data colour assignment, the RGB values are randomly changed on basis of number of distinct classes. Similarly in case of ordinal data, ranking was done on basis of distinct record in data field which used for ranking only.

- How to develop web service that wraps around visualizations service or methods?

From each algorithm, classes were created to design web service. These algorithms were designed to performs specific task and produce visualization rules. The classes were used as service methods or services. All the required methods were combined in one web service, in which some methods were declared as web methods and expose as operations of the web service to users. Java API for XML Web Service (JAX-WS) was used for the creation of web service. Service Interface was generated that expose the required operation to the users. The user consumes web service in their applications, through service

interface. The web service produce visualization rules on basis of input parameters for example data measurement level

- How to structure visualization rules produce by the different web service methods?

Web service methods produce visualization rules or style based on data measurement level. Java String object was used to design xml structure in which these data were stored after required classification approach. We select java because it is the base of almost each type of applications and universally recognized de facto standard for many applications development, such as web-based and mobile based.

- How to develop map file for each data measurement level?

Python mapscirpt API was used to developed program for each data measurement level. Objects were declared in python using this API. These objects have specific task, such mapObj,layerObj,legendObj. Beside API objects parsing algorithms were used for each data level. This method parsed the receive xml document, passed to mapscript program as an argument. After parsing the document, retrieved data were assign to objects in each of called program, such as creation of layer, class expression. Also required expressions were designed using PostgreSQL syntax to retrieve data from the dataset and to define class boundaries. In short after declaring all the required objects, variable, and parsing xml document, data were assigned to variable directly or using loop structure, also WMS request was designed arbitrary, and in last map file are save with name of layer (See appendix-L).

- How to read rules from the service to create map file?

As stated in above answer python parsing library minidom was used to pars xml document. In each method we have function to parse xml document for arbitrary dataset, and retrieve the stored information about style. Parsed information stored in variables within the mapscript method, using loop structure, hereafter assigned to respective object variables to create map file.

- How to associate mapscript methods to web service?

Web service development environment was Java based, and that of map file was Python 2.6 with mapscript API. Both are different technology having different syntax and semantic. We cannot invoke directly one languages method from another. Therefore direct association is hard to establish directly and passed complex structure information as argument to other language platform. There are different options available to associate both technologies.

- Which engine to use for rendering?

We use Map Server 4 Window. This is mapserver setting on window platform. It is easy to install a stable quick working environment for mapserver on windows.

- How to associate all components of VaaS within the framework?

The framework of VaaS (See Figure 5.2) comprise of two main units. The unit which take the decision on the basis of users requests and produce visualization rules after processing the requests. This unit is the web service, return decision in string xml form, also exposes operations to user through Interface and developed in java. The other component takes this xml document and create map file for map server. This component makes use of python mapscirpts. Hence output of one unit digest by other unit to create final visualizations. However these two main units are indirectly associate with each other. The output of first units stored on specified location. This location is passed to other unit, and processed by relevant methods to produce map file. A method was defined that execute python mapscript program from Java web service, and all the steps are executed in the backend of the system. One class method is specified to

stored xml document on drive after classification of data, and other classed method executed to run the python mapscript program, to retrieve this document, parse and create mapfile. These two class method were instantiate in getstyle web method of java web service.

PostgreSQL server is the database server; while GlassFish server was used to deploy web service,; and MapServer for publishing spatial data. These servers are link within the VaaS framework.

We used Java Netbean IDE 7.4 for web service development that consists of DataConnection method responsible to connect with the database located on database server. The link for mapping server was stored in each mapfile, from where content of map file are rendered to user browser using WMS. Hence all components are interconnected, to perform requested visualizations.

- Which choropleth visualizations to use to test the service?

We used various types of choropleth visualizations to prove the concept of Visualizations as a Service (VaaS). We used choropleth visualizations on the basis of data measurement levels. Our main focus was on choropleth visualizations; however VaaS can also render nominal data visualization and produce chorochromatic maps. VaaS can also create proportional symbols visualizations, but still need time for finalization. This proved that VaaS is independent from dataset, and can render numerous types of choropleth visualizations according to user requests.

- Which base map to use to overlay visualizations?

VaaS was tested and visualizations were overlaid on OpenLayers used a base map. For this purpose a simple Graphical user interface (GUI) was designed, using JavaScript and JSP.

- How to overlay arbitrary visualizations on base map?

To overlay arbitrary visualizations on base map, we designed WMS requests dynamic for each request in mapscirpt program. VaaS generate mapfile with name of shapefile dataset. This mapfile name is passed to WMS request each time, which make possible to render arbitrary visualization on OpenLayers.

- What technology could use to realize these implementations?

We use various tools and technology to accomplish this research. For the creation of visualization service, we used Java EE 7 technology. We expose service interface through JAX-WS. The web service was deployed on GlassFish Server. The output of this service is the visualization rules in xml formatted.

To developed map file for mapserver, we used python mapscirpt. Python version 2.6 was installed, as mapsciprt were test and complied against this version. We developed distinct python mapscirpt programs for each data level. Python mapscirpt programs parsed visualizations rules stored in xml document. We used python minidom API to parsed xml document.

MapServer 4 Window-version 3.0.6-MS4W- was used as mapping server. To render final visualization as an image to user we used WMS version 1.1.1.

For spatial dataset generations and maintenance we used PostgreSQL 9.2 server, with PostGIS 2.0. Database was located on PostgreSQL 9.2 server, and shapefile were imported to it.

For the final visualization to user in their browser as an image format, we developed GUI in JSP, JavaScript and HTML.

For data preparations prior to import to database, we used ArcGIS and QGIS. Dataset were prepared in these software before imported to database.

7.9.2 Conclusions and Recommendations

The main objective of this research was to develop loosely coupled visualization framework for the provision of Visualization as a Service (VaaS). By loosely coupled we mean a service to produced visualizations, from arbitrary dataset.

In order to achieve this objective, first algorithms were designed to capture existing choropleth design rules. To this end, we developed a class diagram comprised of all classes including algorithms. Afterwards interface of visualization service was designed to expose functionality of the service to the users. This service was responsible for style/rules regarding choropleth visualizations. These rules were stored as xml document.

From these rules map file was created to produce choropleth visualizations. However for map file creation python map script API was used, to design programs that generate map files from arbitrary xml documents. Also mapscirpt generate WMS request along with map file. Next, map server read map file and produced visualizations in image format as requested.

In our experiments we showed that choropleth maps were produced from arbitrary dataset using framework of VaaS, but that there are limitations. *Firstly,* it is semi-automatic, required data measurement scale from users, to decide visual variable and further classify data and generate visualization rules. *Secondly,* the system used different technology for service development (using java), that responsible for visualization rules and consuming these rules for map file creation (using python mapscript), result in indirect communication between the two units. As visualization rules produced by the service were first saved as a xml document, and afterward retrieved and parsed through python mapscirpt programs. *Thirdly,* the mapfile were saved on map server directory and hereafter read by map server.
Considering the usage of different technology of VaaS and their limitations, this part of the implementation can be improved using one or compatibles technology for both cases, as a result generated rules can pass as an argument, without taking additional step of saving on the specified location for mapscript software. However it depends on maturity of one technology in both disciplines. Third party API is another choice to bring the gap between incompatible programming languages, and to bring us up to language compatibility such as jython.
Mapfile storing on drive has advantage also, for the fulfilment of concurrent requests for similar visualizations, result in rapid responses without iterate the whole process of retrieving data and creation of map file. On the contrary using one map file for multiple visualizations would not reflect dynamic changes of the visualized dataset. However we must to use a map file store on server directory and pass to WMS request as a parameter, because WMS interact with the clients via HTTP protocol using Common Gateway Interface (CGI) and mapserver handled WMS requests using mapserv *CGI* program. It is the *mapserv* that recognise in what way to handle WMS requests. Consequently setting up a WMS server with MapServer involves installing the mapserv CGI program and a setting up a mapfile with proper metadata in it.
At present, an implementation of VaaS results shows data independent visualizations. Producing cartographically valid visualization, for example choropleth map and classification of data will make it possible to designed loosely coupled framework for the provision of Visualization as a Service (VaaS).

Our results prove that visualizations service based on the concept of loosely coupled is successfully applied to predefine problems, such as the semi-automated production of choropleth visualizations and automated selection of visual variable and some classifications approaches. This proved that VaaS is independent from dataset, and can render numerous types of choropleth visualizations according to user requests. However, semantically enables visualizations service are vital to solve more complex problems how, for example, a data measurement scale to be determined automatically.

In conclusion this study reveals that VaaS can be effectively used to develop semi-automated visualizations from arbitrary dataset on demand, and enhance cartographic visualizations functionality. The Interface is exposed to the user and therefore can be consumed by a client application.

Our experiments show that VaaS can semi-automates the process of loosely coupled choropleth map production, but there are risks that a VaaS could produce visualizations that are misinterpreted by humans, due to colour limitations by changing one byte at time. However we also try to develop a colour scheme where we can change all the byte of RGB concurrently, we used Java Colour object, unfortunately the visualizations classes still difficult to differentiate Figure 6.5, and yet certainly wants further research.

Also we observe that the design of VaaS can produce visualization rules for arbitrary dataset, but to produce map file from these rules automatically, without intermediate steps of xml document storage, definitely requires further research.

Likewise we observe that VaaS framework can produce map file for arbitrary dataset, but to send map file to mapserver in API mode with WMS, i.e. without saved with '.map' extension on drive, also need further investigations.

In further work we aim to propose more flexible Interface for other types of visualizations such as proportional symbols map; and provide provision to multiple dataset set-up for example CSV, and also support for the visualization of non-spatial dataset using the concept of geocoding.

REFERENCES

Allamaraju, S. (2010). RESTful Web Services Cookbook (E. M. Treseler Ed. First Edition ed.). United States of America: O'Reilly.

Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2004). Web Services Web Services (pp. 123-149): Springer Berlin Heidelberg.

Andrienko, G., & Andrienko, N. (2002). Computer cartography and cartographic knowledge. Paper presented at the Proceedings: Intercarto 8, International Conference Saint-Petersburg, Russia.

Cerba, O., & Cepicky, J. (2012). Web Services for Thematic Maps. In M. P. Peterson (Ed.), Online Maps with APIs and WebServices (pp. 141-155): Springer Berlin Heidelberg.

Cohen, F. (2002). Understanding web service interoperability,issues in integrating multiple vendor web services implementations. DeveloperWorks. Retrieved from http://www.ibm.com/developerworks/ website: http://www.ibm.com/developerworks/library/ws-inter/ws-inter-pdf.pdf

Dobesova, Z., & Brus, J. (2011). Coping with Cartographical Ontology 11th International Multidisciplinary Scientific Geoconference (pp. 377-384). Sofia: Int Scientific Conference Sgem.

Dobesova, Z., & Brus, J. (2012). Intelligent systems in cartography. In M. V. Koleshko (Ed.), Intelligent Systems (pp. 269-288). Janeza Trdine 9,51000 Rijeka,Croatia: InTech. Retrieved from http://www.intechopen.com/books/intelligent-systems/intelligent-systems-in-cartography.

Ela, & Dramowicz, K. (2004). Choropleth mapping with exploratory data analysis. Retrieved from http://www.directionsmag.com/ website: http://www.directionsmag.com/articles/choropleth-mapping-with-exploratory-data-analysis/123579

Fielding, R., Irvine, U., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., . . . Berners-Lee, T. (1999). Hypertext Transfer Protocol -- HTTP/1.1.

Gillies, S. (un-dated). Pyhton MapScript Appendix. Retrieved 12-12-2013, from http://mapserver.org/mapscript/python.html

GITTA. (2011). Statistics for Thematic Cartography. Retrieved 17-12-2013, from http://www.gitta.info/Statistics/en/html/index.html

Guptill, S. C., & Starr, L. E. (1984). The future of cartography in the information age. In L.E Starr(ed),Computer assisted cartography research and development report ICA Commission Washington:ICA,1-15.

ICA. (1995). 10th General Assembly of the International Cartographic Association. Barcelona, Spain.

IFGI, I. f. G. (2007). Introduction to Digital cartography. Retrieved 25-11-2013, from http://cartography.uni-muenster.de/en/attribute-data

Iosifescu Enescu, I. (2011). Cartographic Web Services. Dissertation for the degree of Doctor of Science. Institute of Cartography and Geoinformation. Swiss Federal Institute of Technology ETH Zurich. Zürich. Retrieved from http://e-collection.library.ethz.ch/view/eth:4541

Jan, B., Zdena, D., Jaromir, K., & Vilem, P. (2010, 24-26 June 2010). Design of intelligent system in cartography. Paper presented at the Roedunet International Conference (RoEduNet), 2010 9th.

Joshua, B. (2005). Python WSRF Programmers' Tutorial. Retrieved 23/11/2013, 2013, from http://acs.lbl.gov/projects/gtg/projects/pyGridWare/doc/tutorial/html/index.html

Kraak, M. J., & Brown, A. (2001). Web cartography : developments and prospects. London etc.: Taylor and Francis.

Kraak, M. J., & Ormeling, F. J. (2010). Catography-visualization of sapatial data. Edinburgh Gate,Harlow,Essex CM20 2JE,england: Pearson Education Limited.

Kraak, M. J., & Ormeling, F. J. (2011). Cartography : visualization of spatial data (Third edition ed.). New York

London: Pearson Education.

Lili, J., Qingwen, Q., & An, Z. (2010, 18-20 June 2010). The thematic mapping system on internet. Paper presented at the Geoinformatics, 2010 18th International Conference on.

Maceachren, A. M., & Kraak, M.-J. (1997). Exploratory cartographic visualization: Advancing the agenda. Computers & Geosciences, 23(4), 335-343. doi: http://dx.doi.org/10.1016/S0098-3004(97)00018-6

Mckenna, J., Fawcett, D., & Butler, H. (Un-dated). An introduction to MapServer. Retrieved 12-01-2014, from http://mapserver.org/introduction.html

Michael, P. P. (2012). Web Services & SOA Principles and Technology: Pearson Education Limited.

Mitchell, T. (2005). Web Mapping Illustrated (S. St.Laurent Ed. First Edition ed.). O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly.

NetBeans. (2013a). Developing JAX-WS Web Service Clients. Retrieved 5/12/2013, from https://netbeans.org/kb/docs/websvc/client.html

NetBeans. (2013b). Web Service Learning Trail. Retrieved 26th, Nov, 2013, from https://netbeans.org/kb/trails/web.html

Neumann, A. (2012). Web Mapping and Web Cartography. In W. Kresse & D. M. Danko (Eds.), Springer Handbook of Geographic Information (pp. 273-287): Springer Berlin Heidelberg.

OGC. (2006). OpenGIS® Web Map Server Implementation Specification.

ORACLE. (2013). The Java EE 6 Tutorial. 500 Oracle Parkway,Redwood City, CA 94065 U.S.A.: Oracle and/or its affiliates.

Pautasso, C., Zimmermann, O., & Leymann, F. (2008). Restful web services vs. "big"' web services: making the right architectural decision. Paper presented at the Proceedings of the 17th international conference on World Wide Web, Beijing, China.

Rautenbach, V., Coetzee, S., & Iwaniak, A. (2013). Orchestrating OGC web services to produce thematic maps in a spatial information infrastructure. Computers, Environment and Urban Systems, 37(0), 107-120. doi: http://dx.doi.org/10.1016/j.compenvurbsys.2012.08.001

Risien, C. M., Allan, J. C., Blair, R., Jaramillo, A. V., Jones, D., Kosro, P. M., . . . Uczekaj, S. A. (2009, 26-29 Oct. 2009). The NANOOS Visualization System: Aggregating, displaying and serving data. Paper presented at the OCEANS 2009, MTS/IEEE Biloxi - Marine Technology for Our Future: Global and Local Challenges.

Roger, L. C. (2008). Building Web Services the REST Way. http://www.xfront.com/files/about.html

Sandoval, J. (2009). RESTful Java Web Services (First Edition ed.). Packt Publishing Ltd.,32 Lincoln Road Olton,Birmingham, B27 6PA,UK: PACKT.

Smith, A. R. (2010, Novermber 15-19,2010). Designing a cartographic ontology for use with expert systems. Paper presented at the A special joint symposium of ISPRS TECHNICAL COMMISSION IV & AutoCarto in conjunction with ASPRS/CaGIS 2010 Fall Specialty Conference, Orlando,Florida.

Stevens, J., Smith, M. J., & Bianchetti, A. R. (2012). Mapping Our Channing World. Retrieved 24/11/, 2013, from https://www.e-education.psu.edu/geog160/c3_p14.html

Sun, X., Shen, S., Leptoukh, G. G., Wang, P., Di, L., & Lu, M. (2012). Development of a Web-based visualization platform for climate research using Google Earth. Computers & Geosciences, 47, 160-168. doi: 10.1016/j.cageo.2011.09.010

Taylor, D., & Almond, S. (1991). Geographic information-systems - the microcomputer and modern cartography Cartographic Journal, 28(2), 269-270.

Vinoski, S. (2008). RESTful Web Services Development Checklist. Internet Computing, IEEE, 12(6), 96-95. doi: 10.1109/MIC.2008.130

Visvalingam, M. (1989). Cartography,GIS and maps in perspective. Cartographic Journal, 26(1), 26-32.

Visvalingam, M. (1990). Trends and concerns in digital cartography Computer-Aided Design, 22(3), 115-130. doi: 10.1016/0010-4485(90)90070-s

W3C. Extensible Markup Language (XML) 1.0 (Fifth Edition),W3C Recommendation 26 November,2008.

Wikipedia. (2013a). Map. Retrieved 23/10/2013, 2013, from http://en.wikipedia.org/wiki/Map

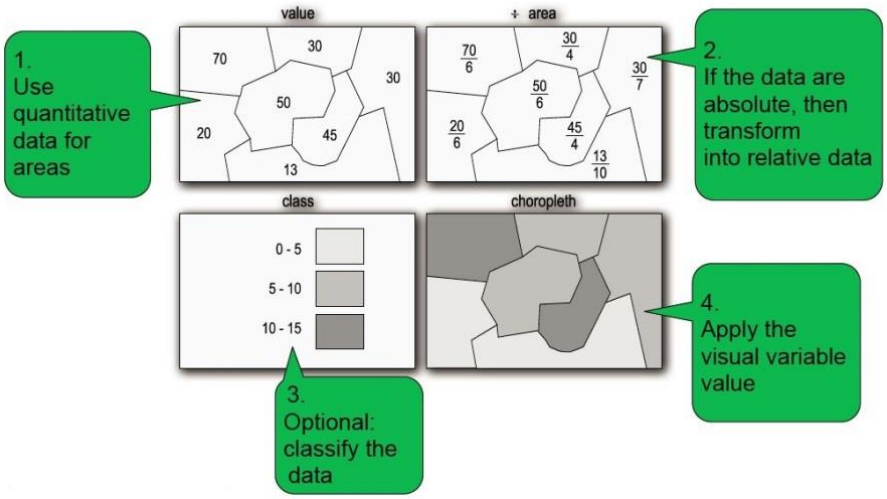
Wikipedia. (2013b). Web Service. Retrieved 12/09/2013, from http://en.wikipedia.org/wiki/Web_service

Wikipedia. (2014). Class Diagram. Retrieved 02-01-2014

Zeng, X., & Zhao, F. (2011, 24-26 June 2011). Ontology Driven Automatic Web Thematic Mapping Based On Web Service Chain. Paper presented at the Geoinformatics, 2011 19th International Conference on.

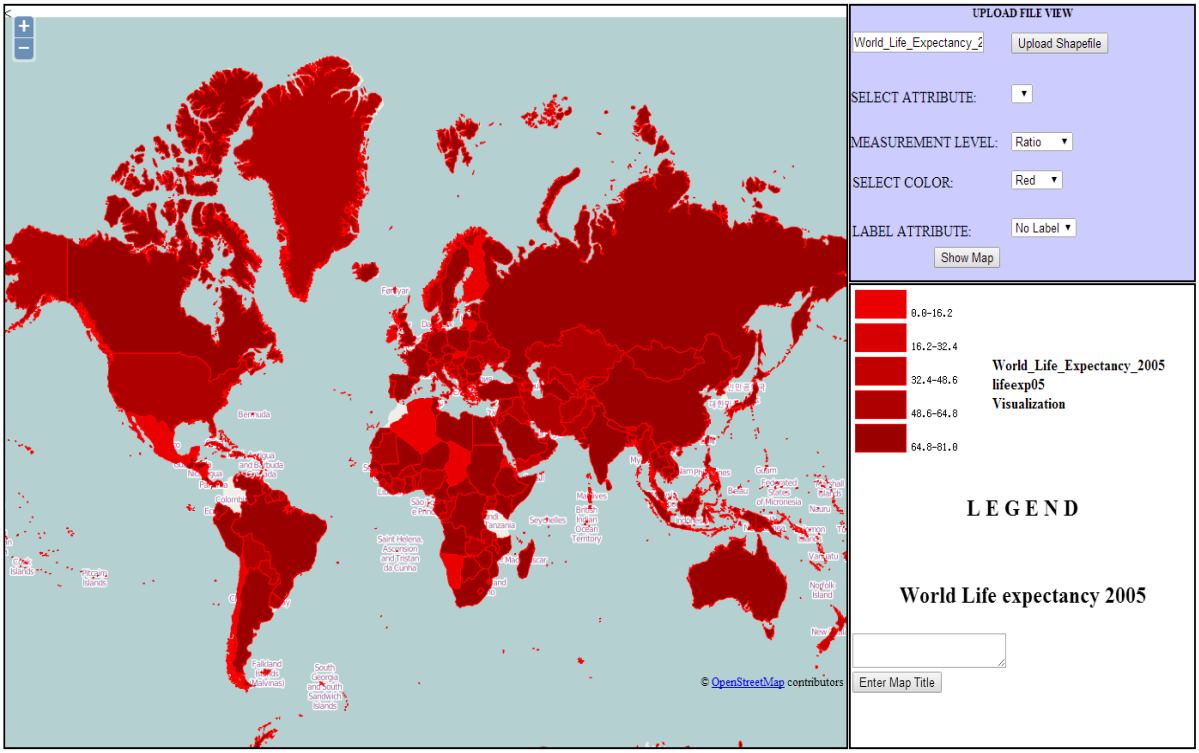
Appendix

A-Choropleth maps production process

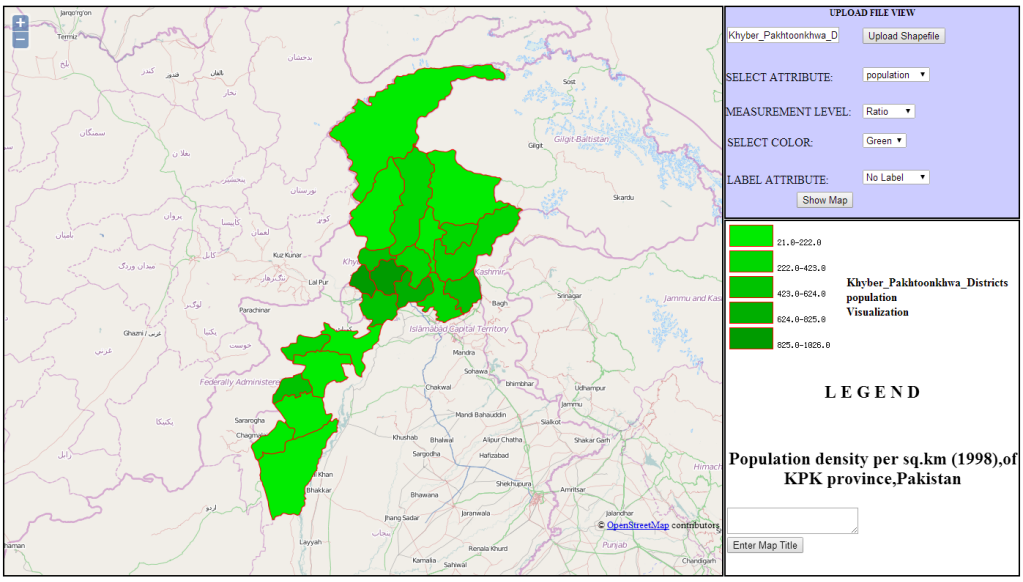


[Source: (Kraak & Ormeling, 2010)]

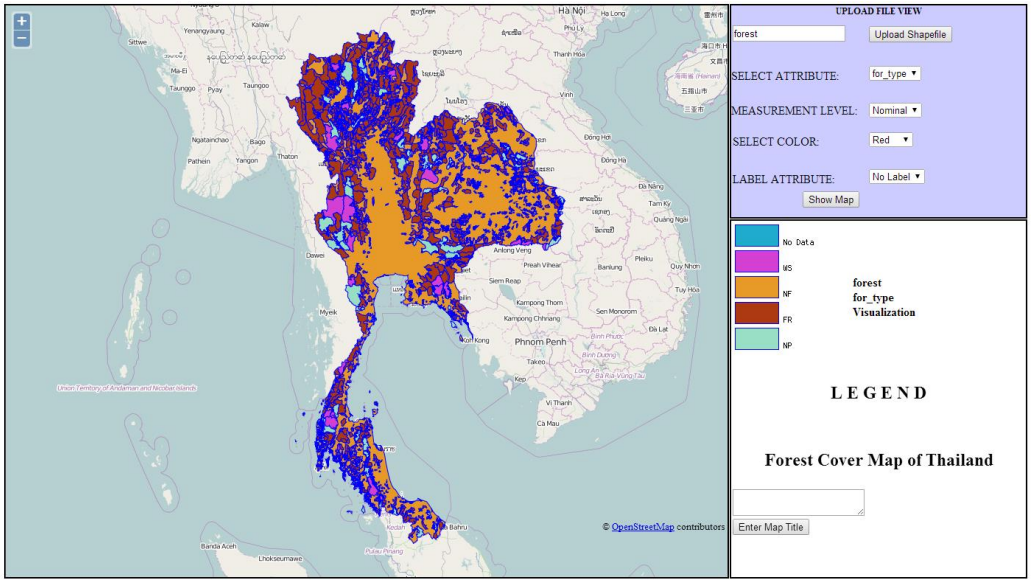
B- RatioANDInterval visualization in red colour (world life expectancy 2005)



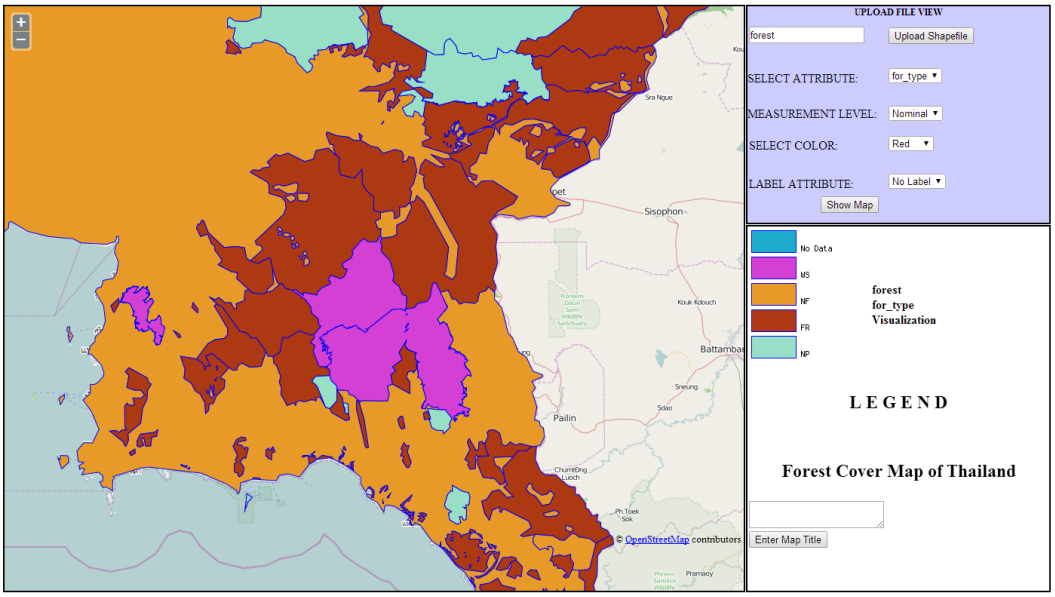
C-RationANDNominal visualization in green colour (Population density)



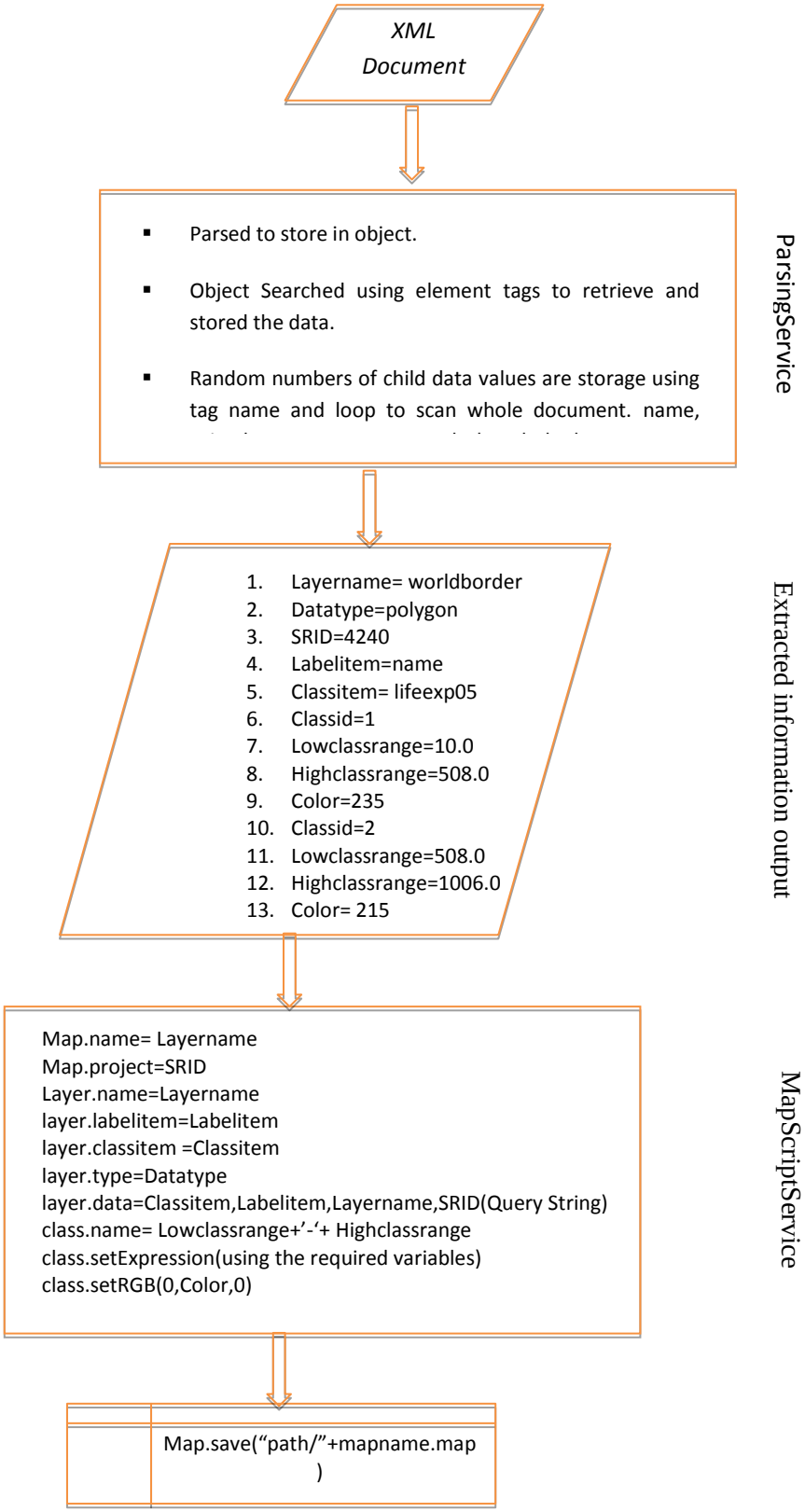
D-Forest land cover visualizations (chorochromatic map), using nominalClassificaiton method



E-Forest Land Cover Visualizations (Zoom in chorochromatic Map)



F- Workflow for the creation of mapfile using MapScript API



```
 *
 * @author sajid
 */
@WebService(serviceName = "DataService")
@Stateless()
public class DataService {

    /**
     * Web service operation
     *
     * @throws java.sql.SQLException
     */
    @WebMethod(operationName = "GetColumnName")
    public String getColumnName(@WebParam(name = "tablename") String tablename) /* throws
SQLException */ {
        String host = "jdbc:postgresql://Localhost:5433/cartography";
        String uName = "postgres";
        String uPass = "admin";
        String finualColumns = "";
        String geomtype = "";
         String newtabname=tablename;
        try {

            Connection con = DriverManager.getConnection(host, uName, uPass);
            Statement statement = con.createStatement();

            ResultSet geomdata = statement.executeQuery("select distinct ST_GeometryType(geom) as geom
from " + tablename);
            while (geomdata.next()) {
                String geoname = geomdata.getString("geom");

                if (!geoname.trim().endsWith("polygon")) {
                    return "Invalid dataset,please upoald valid shapefile with polygon geometry";
                } else {
                    geomtype = "Polygon";
                }

                ResultSet results = statement.executeQuery("SELECT * FROM " + tablename);

                ResultSetMetaData metadata = results.getMetaData();
                int columnCount = metadata.getColumnCount();

                for (int i = 1; i <= columnCount; i++) {
                    String columnName = metadata.getColumnName(i);

                    if (columnName.equalsIgnoreCase("gid")
                            || columnName
                            .equalsIgnoreCase("geom")) {
                        continue;
                    }
                    finualColumns = finualColumns + "," + columnName;
```

```
            }

        }

    } catch (SQLException e) {
    }
    return finualColumns;
}// end of getColumnName method
```

H- Source code of getStyle

```
@WebMethod(operationName = "getStyle")
    public String getStyle(
        @WebParam(name = "columname") String columname,
        @WebParam(name = "measurementtype") String measurementtype,
        @WebParam(name = "tablename") String tablename,
        @WebParam(name = "labelname") String labelname,
        @WebParam(name = "colorlname") String colorlname)
@WebParam(name = "numberofclasses") String numberofclasses)

throws IOException {
    double minvalue = 0.0;
    double minvalueround = 0.0;
    int sridvalue1 = 0;
    double maxvalue = 0;
    double maxvalueround = 0;
    int cvalue = 0;
    String nodata = "";
    //String dclass = "wel come to paksitan";
    String newlabelname = labelname;
    String newcolorname = colorlname;
    if (newlabelname.equalsIgnoreCase("No Label")) {
        newlabelname = "";
    }

    try {
        String host = "jdbc:postgresql://Localhost:5433/cartography";
        String uName = "postgres";
        String uPass = "admin";
        Connection conn = DriverManager.getConnection(host, uName, uPass);
        Statement statement = conn.createStatement();

        //*********************Sql query for minimum value
determinaiton*****************************************************
        ResultSet results2 = statement.executeQuery("select min(" + columname + ") as min FROM " +
tablename);

        while (results2.next()) {
```

```
            minvalue = results2.getInt("min");
            minvalueround = Math.round(minvalue * 100.0) / 100.0;
        }//double roundOff = Math.round(a * 100.0) / 100.0

        //********************Sql query for maximum value
determinaiton*****************************************************
        ResultSet results3 = statement.executeQuery("select max(" + columname + ") as max FROM " +
tablename);
        " + tablename + "where" + columname + "> 0");
        while (results3.next()) {
            maxvalue = results3.getInt("max");
            maxvalueround = Math.round(maxvalue * 100.0) / 100.0;

        }

        ResultSet sriddata = statement.executeQuery("select distinct ST_SRID(geom) as srid2 FROM " +
tablename + " where ST_SRID(geom) is not null");
        while (sriddata.next()) {
            sridvalue1 = sriddata.getInt("srid2");

        }
        ResultSet results6 = statement.executeQuery("select count(DISTINCT(" + columname + "))as cn
FROM " + tablename);
        while (results6.next()) {
            cvalue = results6.getInt("cn");
        }
    } catch (SQLException e) {
        e.getMessage();
    }

    String xmlOutput = "";
    Sridvalue newsridvalue = new Sridvalue();
    int sridval = newsridvalue.sriddata(tablename);

    //******************method for Ratio,Ordinal,Interval classificaiton*****************************
    if (measurementtype.equalsIgnoreCase("interval")
            || measurementtype.equalsIgnoreCase("ratio")) {
        RatioIntervalClassification records1 = new RatioIntervalClassification();

        String records = records1.getClasifications(minvalueround, maxvalueround,newcolorname,5);

        xmlOutput = "\n<Map>\n"
                + " <Layer name=\"" + tablename + "\">\n"
                + " <Type> Polygon </Type>\n</Layer>\n"
                + " <LabelItem name=\"" + newlabelname + "\">\n</LabelItem>\n"
                + " <ClassItem name=\"" + columname + "\">\n</ClassItem>\n"
                + " <SRID name=\"" + sridvalue1 + "\">\n</SRID>\n"
                + records
```

```
            + "</Map>\n";
        System.out.println(xmlOutput);

        xmlToDisk xmlsave = new xmlToDisk();
        xmlsave.xmldata(xmlOutput);


//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 //      Instiate PythonExecute class to execute python
  PythonExecute callpy = new PythonExecute();
        callpy.pycode();

      } else if (measurementtype.equalsIgnoreCase("nominal")) {
        RandomInteger newcol2 = new RandomInteger();
        String nominalcol = newcol2.RgbColor(5);
        String List = getNominalClassification(columname, tablename);
        xmlOutput = "\n<Map>\n"
              + " <Layer name=\"" + tablename + "\">\n"
              + " <Type> Polygon </Type>\n</Layer>\n"
              + " <SRID name=\"" + sridval + "\">\n</SRID>\n"
              + " <LabelItem name=\"" + newlabelname + "\">\n</LabelItem>\n"
              + " <ClassItem name=\"" + columname + "\">\n</ClassItem>\n"
               + List
                + "</Map>\n";
        System.out.println(xmlOutput);

        //pythoncalling();
        xmlToDisk xmlsave2 = new xmlToDisk();
        xmlsave2.xmldata(xmlOutput);

        NominalMapScript newjava2 = new NominalMapScript();
        newjava2.pycode();

      } else if (measurementtype.equalsIgnoreCase("ordinal")) {
        OrdinalClassification records1 = new OrdinalClassification();
        String List = records1.getOrdinalClassification(columname, tablename);

//Ordinal classifications xml structure
        xmlOutput = "\n<Map>\n"
              + " <Layer name=\"" + tablename + "\">\n"
              + " <Type> Polygon </Type>\n</Layer>\n"
              + " <SRID name=\"" + sridvalue1 + "\">\n</SRID>\n"
              + " <LabelItem name=\"" + newlabelname + "\">\n</LabelItem>\n"
              + " <ClassItem name=\"" + columname + "\">\n</ClassItem>\n"
              + " <Colortype name=\"" + newcolorname + "\">\n</Colortype>\n"
              + List
              + "</Map>\n";
        System.out.println(xmlOutput);
```

```
            xmlToDisk xmlsave3 = new xmlToDisk();
            xmlsave3.xmldata(xmlOutput);
             OrdinalMapScript newjava = new OrdinalMapScript();
            newjava.pycode();

        }
        return xmlOutput;
    }// end of getStyle method

// This classifiation is ued for nominal data type to prepare chorochromatic e.g. Landcover map
//(Forest,Agriculture,Range Land,Barand Land etc)
    private String getNominalClassification(String columname2, String tablename2) {

        String dclass = "";
        String newrecord = "";
        String host = "jdbc:postgresql://Localhost:5433/cartography";
        String uName = "postgres";
        String uPass = "admin";

        try {
            Connection con = DriverManager.getConnection(host, uName, uPass);
            Statement statement = con.createStatement();
        tablename2 +"where ST_SRID(geom) is not null");
            ResultSet results7 = statement.executeQuery("select DISTINCT(" + columname2 + ") as dt
FROM " + tablename2);
            while (results7.next()) {
                dclass = results7.getString("dt");
                int red = randomGenerator.nextInt(255);
                int green = randomGenerator.nextInt(255);
                int blue = randomGenerator.nextInt(255);


                newrecord += "<classes>\n"
                        + "<rank>" + dclass + "</rank>\n"
                        + "<red>" + red + "</red>\n"
                        + "<green>" + green + "</green>\n"
                        + "<blue>" + blue + "</blue>\n</classes>\n";
            }
        } catch (SQLException e) {

        }
        return newrecord;
    }


} // end of visualization service
```

F-Source Code of RatioANDIntervalClassification

```
package vaas;

/**
 *
 * @author sajid
 */
public class RatioIntervalClassification {

    public String getClasifications(double min, double max, int numberOfClasifications) {
        double resutlValue = (max - min) / numberOfClasifications;
        int constantValue = 20;
        String classifications = "";
        for (int i = 1; i <= numberOfClasifications; i++) {
            double rangeMax = min + resutlValue;
            double rangeMin = rangeMax - resutlValue;
            double rangeMinr = Math.round(rangeMin * 100.00) / 100.00;
            double rangeMaxr = Math.round(rangeMax * 100.00) / 100.00;
            int rangeColor = 255 - (constantValue * i);
            classifications += "<classes Id=\"" + i + "\">\n"
                    + " <minBoundary>" + rangeMinr + "</minBoundary>\n"
                    + " <maxBoundary>" + rangeMaxr + "</maxBoundary>\n"
                    + " <color> " + rangeColor + "</color>\n</classes>\n";

            min = rangeMax;
        }
        return classifications;
    }
    public static void main(String args[]) {
        RatioIntervalClassification c = new RatioIntervalClassification();
        String classdata = c.getClasifications(10, 2500, 5);
        System.out.println(classdata);

    }
}
```

G-Source code of HarmonicSeries classification

```
package vaas;
//import vaas.OrdinalClassification;
import org.python.modules.math;

/**
 *
 * @author sajid
 */
public class HarmonicSeries {

    public String getClasifications(double min, double max, int numberOfClasifications) {
        double rmax = 0;
        double rmin = 0;

        int constantValue = 20;
```

```
        String classifications = "";
        rmin = (double) 1 / min;
        rmax = (double) 1 / max;
        double c = Math.abs(rmax - rmin) / numberOfClasifications;
        for (int i = 1; i <= numberOfClasifications; i++) {

            double rangeMax = Math.abs(rmax - c);
            double rangeMin = Math.abs(rangeMax - c);

            double rangeMinr =Math.round( (1 / rangeMax *100.00))/100.00;
            double rangeMaxr = Math.round( (1 / rangeMin*100.00))/100.00;

            int rangeColor = 255 - (constantValue * i);
            classifications += "<classes Id=\"" + i + "\">\n"
                    + " <minBoundary>" + rangeMinr + "</minBoundary>\n"
                    + " <maxBoundary>" + rangeMaxr + "</maxBoundary>\n"
                    + " <color> " + rangeColor + "</color>\n</classes>\n";
            rmax -= c;
        }
        return classifications;

    }
```

H-Source Code of NOMINAL classification

```
private String getNominalClassification(String columname2, String tablename2) {
            String dclass = "";
        String newrecord = "";
        String host = "jdbc:postgresql://Localhost:5433/cartography";
        String uName = "postgres";
        String uPass = "admin";

        try {
            Connection con = DriverManager.getConnection(host, uName, uPass);
            Statement statement = con.createStatement();
FROM"+ tablename2 +"where ST_SRID(geom) is not null");

            ResultSet results7 = statement.executeQuery("select DISTINCT(" + columname2 + ") as dt
FROM " + tablename2);

            while (results7.next()) {
                dclass = results7.getString("dt");
                int red = randomGenerator.nextInt(255);
                int green = randomGenerator.nextInt(255);
                int blue = randomGenerator.nextInt(255);

                newrecord += "<classes>\n"
                        + "<rank>" + dclass + "</rank>\n"
                        + "<red>" + red + "</red>\n"
                        + "<green>" + green + "</green>\n"
```

```
                + "<blue>" + blue + "</blue>\n</classes>\n";
        }
    } catch (SQLException e) {

    }
    return newrecord;
```

I-Source code of ORDINALclassification

```
public class OrdinalClassification {


  public String getOrdinalClassification(String columname2, String tablename2) {
        String dclass = "";
    String newrecord = "";
    String host = "jdbc:postgresql://Localhost:5433/cartography";
    String uName = "postgres";
    String uPass = "admin";
    int red = 0;
    int cvalue = 0;
    try {
      Connection con = DriverManager.getConnection(host, uName, uPass);
      Statement statement = con.createStatement();
    FROM"+ tablename2 +"where ST_SRID(geom) is not null");
      Vegetation_Height");
       ResultSet results6 = statement.executeQuery("select COUNT(" + columname2 + ") as cn FROM
" + tablename2);
      while (results6.next()) {
        cvalue = results6.getInt("cn");
      }
            ResultSet results7 = statement.executeQuery("select DISTINCT(" + columname2 + ") as
dt FROM " + tablename2);
      for (int i = 1; i <=  cvalue; i++) {
        red = 255;
                while (results7.next()) {
```

```
                dclass = results7.getString("dt");
                newrecord += "<classes>\n"
                        + "<order>" + dclass + "</order>\n"
                        + "<color>" + red + "</color>\n"
                        + "</classes>\n";
                red -= 50;

            }
          }
        } catch (SQLException e) {

        }
        return newrecord;
    }
```

J-RGB Color schem

```
package vaas;

import java.awt.Color;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

/**
 *
 * @author sajid
 */
public class ColorScheme {
 String bb="";
    public List<Color> calculateShades(Color baseColor, int numberShades) {
        //decompose color into RGB
        int redMax = baseColor.getRed();
        int greenMax = baseColor.getGreen();
        int blueMax = baseColor.getBlue();

        //Max color component in RGB
        final int MAX_COMPONENT = 240;

        //bin sizes for each color component
        int redDelta = (MAX_COMPONENT - redMax) / numberShades;
        int greenDelta = (MAX_COMPONENT - greenMax) / numberShades;
        int blueDelta = (MAX_COMPONENT - blueMax) / numberShades;
```

```
        List<Color> colors = new ArrayList<Color>();

        int redCurrent = redMax;
        int greenCurrent = greenMax;
        int blueCurrent = blueMax;

        //now step through each shade, and decrease darkness by adding color to it
        for (int i = 0; i < numberShades; i++) {

            //step up by the bin size, but stop at the max color component (255)
            redCurrent = (redCurrent + redDelta) < MAX_COMPONENT ? (redCurrent + redDelta) :
MAX_COMPONENT;
            greenCurrent = (greenCurrent + greenDelta) < MAX_COMPONENT ? (greenCurrent +
greenDelta) : MAX_COMPONENT;
            blueCurrent = (blueCurrent + blueDelta) < MAX_COMPONENT ? (blueCurrent + blueDelta) :
MAX_COMPONENT;

            Color nextShade = new Color(redCurrent, greenCurrent, blueCurrent);



            colors.add(nextShade);

        }

        return colors;
    }

}
```

J-Source code of XMLData storage Service

```
package vaas;
public classXMLData {

   public void xmldata(String xmlrecord){
      File file = new File("D:\\pythonmapfile\\mapcreation\\mapscriptmap\\newfile.xml");
            String content = xmlrecord;

            try (FileOutputStream fop = new FileOutputStream(file)) {

               // if file doesn't exists, then create it
               if (!file.exists()) {
                  file.createNewFile();
               }
               // get the content in bytes
               byte[] contentInBytes = content.getBytes();

               fop.write(contentInBytes);
               fop.flush();
               fop.close();
               System.out.println("Done");
            } catch (IOException e) {
               e.printStackTrace();
            }
```

```
    }
    public static void main(String[] args) throws IOException {
        xmlToDisk xmldatasave=new xmlToDisk();


    }
}
```

K-1-Source code of executing MapScriptService from Java

```
/**
*
 * @author sajid
 */
public class Execute {

    /**
     * Executes a command (by java.lang.Runtime.exec). Also takes care of the
     * command-line parameters to be passed and captures the standard-output
     * generated.
     *
     * @param command
     *            Path to the executable file
     * @param parameters
     *            Command-line parameters to be passed
     * @return The standard-output that is generated by executing the command,
     *         in an array of strings, one string per line of output.
     */
    public static List<String> executeCommand(String command,
            List<String> parameters) throws IOException,
            InterruptedException {
        String commandLine = quoteIfNeeded(command);
        int lastSlash = commandLine.lastIndexOf('\\');
        String workingDir = (lastSlash>=0)?commandLine.substring(0, lastSlash):".";

        if (parameters != null) {
            for (int i = 0; i < parameters.size(); ++i)
                commandLine += " " + quoteIfNeeded(parameters.get(i));
        }

        Process p = Runtime.getRuntime().exec(commandLine, null,
                new File(workingDir));
        BufferedReader br = new BufferedReader(new InputStreamReader(p
                .getErrorStream()));
        List<String> output = new ArrayList<String>();
        String line = br.readLine();
        while (line != null) {
            output.add(line);
            line = br.readLine();
        }
        return output;
    }

    private static String quoteIfNeeded(String s) {
```

```
        final String forbiddenChars = " \""; // space, quote
        for (int i = 0; i < forbiddenChars.length(); ++i) {
            char c = forbiddenChars.charAt(i);
            if (s.indexOf(c) >= 0)
                return "\"" + s + "\"";
        }
        return s;
    }
}
```

K-2-Source code of executing MapScriptService from Java

```
/**
 *
 * @author sajid
 */
public class PythonExecute {
        public void pycode(){
        try {
            List<String> parameters = new ArrayList<String>();
            parameters.add("path to MapScriptService\\mapscriptservice.py");
            List<String> output = Execute.executeCommand("C:\\Python26\\python.exe", parameters);
            if (output != null)
                for (String out : output)
                    System.out.println(out);

        } catch (Exception e) {

            System.out.println(e.getMessage());
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        PythonExecute pythex=new PythonExecute();
        pythex.pycode();

    }

}
```

L-Source code of MapScriptService

```
'''
Created on Jan 29, 2014

@author: sajid
'''

from xml.dom import minidom
from xml.etree import ElementTree as ET
import urllib
import mapscript
```

```
map1 = mapscript.mapObj()
weboj = mapscript.webObj()
layer = mapscript.layerObj(map1)
map1.setSize(600,800)
a=("C:/ms4w/apps/thailand/fonts/fonts.list")
map1.setFontSet(a)

def getlayername(xmldocument):
        source=url
        lname=''
        xmldoc = minidom.parse(source)
        xmldocelement = xmldoc.getElementsByTagName('Layer') # read the content of
xml tag
        xmlchildn = xmldoc.childNodes

        for node in xmldocelement:
            if node.attributes.has_key('name'):

                lname = node.attributes['name'].value
                LayerName = lname
                map1.name = LayerName

                return LayerName

def MapScriptService(xmldocument):
        source=url

        count = 0
        xmldoc = minidom.parse(source)
        xmldocelement = xmldoc.getElementsByTagName('classes')
        xmldocelement3 = xmldoc.getElementsByTagName('ClassItem')
        Layerxml = xmldoc.getElementsByTagName('LabelItem')
        srid2 = xmldoc.getElementsByTagName('SRID')
        colortype = xmldoc.getElementsByTagName('Colortype')
        for node in colortype:
            if node.attributes.has_key('name'):
                newcolortype = node.attributes['name'].value

        for node in Layerxml:
            if node.attributes.has_key('name'):
                LabelItem = node.attributes['name'].value
                if LabelItem=="":
                    not layer.labelitem
                else:
                    layer.labelitem=LabelItem


        for node in srid2:
            if node.attributes.has_key('name'):
                sridvalue = node.attributes['name'].value


        for node in xmldocelement3:
            if node.attributes.has_key('name'):
                ClassItem = node.attributes['name'].value

        xmldocelement1 = xmldoc.getElementsByTagName('Layer') # read the content of
xml tag


        for node in xmldocelement1:
            if node.attributes.has_key('name'):

                lname = node.attributes['name'].value
```

```
        numberOfClasses=len( xmldocelement)
        layer.name = lname


        layer.classitem =ClassItem

        layer.type =mapscript.MS_LAYER_POLYGON
        layer.status = mapscript.MS_DEFAULT
        layer.connectiontype=mapscript.MS_POSTGIS
        layer.connection="user = postgres password=admin dbname=cartography
host=localhost port=5433"
        classa=ClassItem+"\tAS\t"+ClassItem

        if LabelItem=="":
            commaop=","
            tabop="\tAS\t"
            labelc=LabelItem+LabelItem
            not labelc
            not commaop
            not tabop
        else:
            labelc=(","+LabelItem+"\tAS\t"+LabelItem)


        layer.data="ogc_geom FROM ( SELECT geom AS ogc_geom, gid AS ogc_id,"+
classa + labelc + "\tFROM public." +lname +" ) AS query USING UNIQUE ogc_id USING
srid="+sridvalue+""
        layer.metadata.set('ows_title' ,lname)
        map1.web.metadata.set('ows_srs','epsg:'+sridvalue+'\t''epsg:3857')


        for node in xmldocelement:
                minBoundary= node.getElementsByTagName('minBoundary')
                fristMinBoundary=minBoundary[0].firstChild.data

                maxBoundary= node.getElementsByTagName('maxBoundary')
                fristMaxBoundary=maxBoundary[0].firstChild.data

                color= node.getElementsByTagName('red')
                redColor=color[0].firstChild.data
                resultsred = int(redColor)


                color2= node.getElementsByTagName('green')
                greenColor=color2[0].firstChild.data
                resultsgreen = int(greenColor)

                color3= node.getElementsByTagName('blue')
                blueColor=color2[0].firstChild.data
                resultsblue = int(blueColor)

                #// To create arbitrary classes
                classnew=mapscript.classObj(layer)
                #filter2 = (("([" + ClassItem +"]\t"+ "<=0"  +"\tOR\t["+ ClassItem +
"]= NULL)")

                #classnew.name =fristMinBoundary +'-'+ fristMaxBoundary
                classnew.name = fristMaxBoundary+'-'+ fristMinBoundary

                #filter1 = (("([" + ClassItem +"]\t"+ ">=" + fristMinBoundary
+"\tAND\t["+ ClassItem + "]<=" + fristMaxBoundary +")")
                filter1 = ("([" + ClassItem +"]\t"+ "<=" + fristMinBoundary
+"\tAND\t["+ ClassItem + "]>=" + fristMaxBoundary +")")
```

```
                #filter2 = ("([" + ClassItem +"]\t"+ "<=0"  +"\tOR\t["+ ClassItem +
"]= NULL)")

                classnew.setExpression(filter1)
                newcolorset = mapscript.styleObj(classnew)

                newcolorset.color.setRGB(resultsred,resultsgreen,resultsblue)

newcolorset.outlinecolor.setRGB(resultsred,resultsgreen,resultsblue)


                classnew.label.color.setRGB(0,0,0)
                classnew.label.type=mapscript.MS_TRUETYPE
                classnew.label.font = 'arial'

                classnew.label.size=12
                classnew.label.position = mapscript.MS_AUTO
                classnew.label.partials=False
                classnew.label.buffer=4
                classnew.label.antialias = mapscript.MS_TRUE

     //wms request generation

mapname =getlayername('D:/pythonmapfile/mapcreation/mapscriptmap/newfile.xml')
dmapfile=mapname
map1.web.metadata.set('ows_enable_request','*')
map1.web.metadata.set('map','C:/ms4w/apps/thailand/'+dmapfile+'.map')
map1.web.metadata.set('ows_schemas_location','http://schemas.opengeospatial.net')
map1.web.metadata.set('ows_title', dmapfile +'\tWMS')
map1.web.metadata.set('ows_onlineresource','http://Localhost:9090/Apachi/cgi-
bin/mapserv.exe?map=C:/ms4w/apps/thailand/'+dmapfile+'.map&')

map1.web.metadata.set('wms_feature_info_mime_type','text/plain')
map1.web.metadata.set('wms_feature_info_mime_type','text/html')
map1.web.metadata.set('wms_server_version','1.1.1')
map1.web.metadata.set('wms_formatlist','image/png,image/gif,image/jpeg')
map1.web.metadata.set('wms_format','image/png')
map1.web.imagepath=("C:/Users/sajid/AppData/Local/Temp/")
map1.web.imageurl=("C:/Temp/")


if __name__ == '__main__':
    MapScriptService('path to xml document/XMLDocument.xml')
    mapname =getlayername('path to xml document/XMLDocument.xml')
    dmapfile1=map1.save("path to save mapfile/" + mapname +".map")
```