AN ONLINE INTERACTIVE MULTIPLE-PERSPECTIVES MAP SYSTEM

By

JING HUO

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2002
Dedicated to my husband, Jinping Zheng, and my son, Mumu
ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Dr. Richard Newman, for his excellent advice and guidance throughout my whole thesis work. He led me into a good topic and an interesting research area. He is a good advisor. I also thank Dr. Abdelsalam Helal and Dr. Randy Chow for serving as my master’s thesis committee.

I would like to thank Tom and Tingting, who worked so hard and so carefully to proofread my thesis. Their love, encouragement, and help will keep in my heart forever.

I would like to thank Meifang Tan, who is my thesis partner and my friend. She gave me so much help and many happy times.

I would like to thank Lijun Chen and Dongqing Xie for their valuable advice and help.

I would like to thank my parents and my sisters for their endless love.

Finally, I would like to thank my husband, who is always there to support me.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>ix</td>
</tr>
<tr>
<td>CHAPTERS</td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>Description of the Problems</td>
<td>1</td>
</tr>
<tr>
<td>Overview of OIMPMS</td>
<td>4</td>
</tr>
<tr>
<td>Information Visualization Product and Development</td>
<td>6</td>
</tr>
<tr>
<td>Organization of Chapters</td>
<td>8</td>
</tr>
<tr>
<td>2. UNDERLYING TECHNOLOGIES</td>
<td>9</td>
</tr>
<tr>
<td>Overview of Java Technology</td>
<td>9</td>
</tr>
<tr>
<td>Basic Structure Paradigm of Web-Based System</td>
<td>11</td>
</tr>
<tr>
<td>Database Access Solutions</td>
<td>12</td>
</tr>
<tr>
<td>Two-Tier Connection Structure</td>
<td>12</td>
</tr>
<tr>
<td>Three-Tier Connection Structure</td>
<td>13</td>
</tr>
<tr>
<td>Object Serialization</td>
<td>16</td>
</tr>
<tr>
<td>Summary</td>
<td>17</td>
</tr>
<tr>
<td>3. SOFTWARE FUNCTION</td>
<td>19</td>
</tr>
<tr>
<td>Visualization Modes</td>
<td>19</td>
</tr>
<tr>
<td>Software Functions</td>
<td>20</td>
</tr>
<tr>
<td>Loading an Application</td>
<td>20</td>
</tr>
<tr>
<td>Creating a New Application</td>
<td>21</td>
</tr>
<tr>
<td>Modifying an Application Setup</td>
<td>23</td>
</tr>
<tr>
<td>Saving an Application</td>
<td>23</td>
</tr>
<tr>
<td>Deleting an Application</td>
<td>23</td>
</tr>
<tr>
<td>Adding an Entity Item</td>
<td>24</td>
</tr>
<tr>
<td>Inspecting, Deleting, and Modifying Entity Items</td>
<td>26</td>
</tr>
<tr>
<td>Moving an Entity Item (MODE1)</td>
<td>26</td>
</tr>
</tbody>
</table>
Inserting an Entity Item (MODE2) ................................................................. 26
Dragging and Moving a Borderline (MODE2) ............................................... 27
Summary ........................................................................................................... 27

4. SYSTEM DESIGN AND IMPLEMENTATION ................................................. 28

System Design ................................................................................................. 28
Java Applet ..................................................................................................... 28
Communication Link ...................................................................................... 29
Database Design ........................................................................................... 30
Java Servlet .................................................................................................. 32
Object Design ............................................................................................... 33
System Implementation .................................................................................. 34
System Environment Setup .......................................................................... 34
Software Installation ..................................................................................... 35
Summary ........................................................................................................ 35

5. TESTS AND RESULTS .................................................................................. 37

State-Based Testing ....................................................................................... 37
Test Implementations and Results ............................................................... 39
Summary ........................................................................................................ 42

6. CONCLUSIONS ............................................................................................ 43

Summary of OIMPMS .................................................................................. 43
Some Suggestions for Improving OIMPMS ...................................................... 44
Summary ........................................................................................................ 45

LIST OF REFERENCES ..................................................................................... 46

BIOGRAPHICAL SKETCH ............................................................................... 48
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1. Typical data classification support for OIMPMS</td>
<td>5</td>
</tr>
<tr>
<td>4-1. Tomcat directory structure table</td>
<td>34</td>
</tr>
<tr>
<td>5-1. Sample of the Test Data used in Example 1</td>
<td>39</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1. Basic Structure Paradigm of Web-Based System</td>
<td>11</td>
</tr>
<tr>
<td>2-2. A Two-Tier Connection Structure of Java Applet and Database</td>
<td>12</td>
</tr>
<tr>
<td>2-3. Three-Tier Connection Structure (HTTP) of Java Applet and Database</td>
<td>14</td>
</tr>
<tr>
<td>2-4. Three-Tier Connection Structure (RMI) of Java Applet and Database</td>
<td>15</td>
</tr>
<tr>
<td>2-5. Three-Tier Connection Structure (Socket) of Java Applet and Database</td>
<td>16</td>
</tr>
<tr>
<td>2-6. Object Serialization Process Example Code</td>
<td>17</td>
</tr>
<tr>
<td>3-1. Loading an Application</td>
<td>20</td>
</tr>
<tr>
<td>3-2. Application Setup</td>
<td>22</td>
</tr>
<tr>
<td>3-3. Adding an Entity Item</td>
<td>24</td>
</tr>
<tr>
<td>3-4. Inspecting and Modifying an Entity Item</td>
<td>25</td>
</tr>
<tr>
<td>4-1. Database Table Structure</td>
<td>31</td>
</tr>
<tr>
<td>4-2. An Excerpt From “server.xml” File</td>
<td>35</td>
</tr>
<tr>
<td>5-1. System State Diagrams</td>
<td>38</td>
</tr>
<tr>
<td>5-2. Visualization Result – Pie Chart of Example 1</td>
<td>40</td>
</tr>
<tr>
<td>5-3. Visualization Result – Column Chart of Example 1</td>
<td>41</td>
</tr>
<tr>
<td>5-4. Visualization Result of Example 2</td>
<td>42</td>
</tr>
</tbody>
</table>
AN ONLINE INTERACTIVE MULTIPLE-PERSPECTIVES MAP SYSTEM

By

Jing Huo

August 2002

Chairman: Dr. Richard Newman
Major Department: Computer and Information Science and Engineering

This thesis presents the design and implementation of an Online Interactive Multiple-Perspectives Map System (OIMPMS). OIMPMS is an online interactive information visualization application system that generates multiple-perspectives maps. It supports visualized presentation of charts derived from category data. The charts are distributed on a map according to the geographic locations corresponding to the underlying data. The system is interactive in two respects: one is it supports for the interface friendly visualized input of data for the user-specified categories and the other is it supports for selection by the user from among multiple visualization application modes and chart types for inclusion. Currently OIMPMS supports three chart types and two application modes for building multiple-perspectives maps that differ with respect to chart positioning and geographic region demarcation.

OIMPMS is built on a three-tier connection structure with Java Applets on the client, Java Servlets on the web server, and the database and files on separate database servers
and file servers. The object serialization mechanism in Java is exploited in OIMPMS based on the successful development and implementation of a method for overcoming the typical limitations of this connection structure. OIMPMS has demonstrated the feasibility of an interactive information visualization solution and conceptual design for the online generation of multiple-perspectives maps.
CHAPTER 1
INTRODUCTION

This chapter begins with a description of the problems encountered in the
development of online interactive visualization applications. This description is followed
by an overview of the Online Interactive Multiple-Perspectives Map System (OIMPMS),
which is a computer application system developed to address these problems. Next, some
information about visualization application products will be introduced. The organization
of chapters in this thesis is presented at the end of this chapter.

Description of the Problems

The human eye interprets visual data much faster and more effectively than text. A
picture or diagram can be assimilated at a glance, and our understanding of the subject is
greatly enhanced by the image [1]. Most of us are familiar with the old adage, "A picture
is worth a thousand words"--or its modern version, "An image is worth a megabyte." The
rapid development of computer technology has produced new methods for
communicating information in visual or graphical form. New forms of information
visualization technology that gradually emerged within this context have produced some
widely used applications, especially in the fields of geographic information systems,
medical diagnostic systems, spatial technology information systems, etc.

Visualization is the graphical presentation of information. It combines aspects of
scientific visualization, human-computer interaction, data mining, imaging and graphics
[2]. Information visualization is a process that transforms data, information, and
knowledge into a form that relies on the human visual system to perceive its embedded
information. Its goal is to enable the viewer to observe, understand, and make sense of
the information [3]. An information visualization application can provide the viewer with
a qualitative understanding of the information content and enable the user/viewer to get
information quickly, gain new insights into data and make decisions in a relatively short
time.

Visualization software is designed to meet the needs of two groups of users. One
group consists of those interested in developing their own visualization environments.
For these users--often researchers, but sometimes engineers as well--what they need is a
strong data model to incorporate data of widely disparate types in one cohesive
environment. The other group wants completed, specialized, and final visualization
applications that allow them to view and analyze results immediately. For example,
engineers often need visualization software to provide them with a way to investigate the
potential of new designs by dynamically altering and tuning simulation parameters. By
doing this, they can interactively monitor the progress of a simulation and terminate poor
design directions [4].

For the second group, visualization systems must focus on very specialized
disciplines, with very specific problems and data sources. The reason for this requirement
is that information in these disciplines covers a wide range of forms and includes
extensive connotations. In actual applications, information may include such forms as
data, processes, relations, or concepts that represent various meanings. The form of
information and the characteristics of data determine the graphics type required to capture
and communicate the meanings within the information and data. The best graphical
representation for the contents of stock exchange data, for example, is the line chart,
while the pie chart best represents the meaning of data in the form of percentage composition. The restricted application of specialized visualization software therefore requires that a different visualization application be developed for every field, every specific problem.

For both user groups, data classification is also a crucial task to perform in the visualization process because the type and structure of the data define the set of graphical mappings that can be performed on it.

Just as in any other computer application system, interactivity is a very important aspect of an information visualization application or system. It is interactivity that provides users and developers with the invaluable capacity and freedom to explore a broad range of functions within a computer application system. Object-oriented programming technology lets users and developers create more sophisticated and interactive tools for visualization than were available in the past. In this technology, the visualization consists of a series of graphics objects. Object-oriented programming methods provide a flexible and elegant means for achieving "visual" representations based on graphics objects that can be rapidly redrawn without being recalculated [5].

Today the World Wide Web (WWW) has become the single most important and extensive information space, containing millions of pages. Web-based information visualization is undoubtedly a significant, promising technique, receiving constantly increasing attention. Online information visualization applications that use the browser as a user side platform and the web server as an information source support are proving highly advantageous.
Most web-based visualization applications use the Java programming language. Java is an object-oriented language with many features that make Java-based online visualization applications more flexible as well as easier to use and to produce.

In response to the above problems and potentials of online interactive applications for information visualization, OIMPMS was designed and developed, trying to provide an online interactive, specialized visualization application for multiple-perspectives map.

Overview of OIMPMS

Maps usually communicate a single message statically. During the past decade, people began to explore visualization applications based on maps that present multiple perspectives of information dynamically [6]. Assume that we have a group of statistical data from an area (such as marine or mineral resource statistics for the State of Florida, for instance) and want to represent this group of data graphically. Once we have chosen a type of graphics, such as a bar chart, we can produce these graphics in Excel or with other available drawing tools. We can increase the information by placing charts representing data from multiple states on a map of the United States, but a dynamic method for constructing and positioning graphics requires a more sophisticated visual representation system.

We developed an application to translate a group of data from Florida, for instance, into a graphical image of a particular type and then to place that image on a map of the United States, centering on a coordinate point within the boundaries of the State of Florida. If we have mineral resource statistics for all the states in the USA, the application produces a dynamic, multiple-perspectives map with a graphic for each state representing its group of data. Each of these graphics is centered on a coordinate within the boundaries of the corresponding state. Thus we know which geographic site the data
applies to based on the location where its chart is drawn. Moreover, we know the 
qualitative meaning of different categories of data based on the size, color, and 
description of the chart.

OIMPMS is an information visualization application system featuring online, 
interactive generation of multiple-perspectives maps. The system supports visualized 
presentation of charts derived from category data (such as population projections by the 
Census Bureau). The charts are distributed on a map according to the geographic 
locations corresponding to the underlying data. It also has achieved the interactivity 
required of flexible and dynamic information visualization applications. Interactivity has 
a dual meaning here: one corresponds to the capacity for interface-friendly visualized 
input of data for the user-specified category and the other is its support for selection by 
the user from among multiple visualization application modes and chart types for 
inclusion for the data groups. This latter feature allows a multiple-perspectives map of 
specific mode and specific chart type to be generated according to the specifications of 
the user.

The typical data classification support for OIMPMS (Table 1-1) is the underlying n×p 
table, where we store n data points (rows), each with p variables (columns).

Table 1-1. Typical data classification support for OIMPMS.

<table>
<thead>
<tr>
<th>Location Coordinate1 (X or Y1)</th>
<th>Location Coordinate2 (Y or Y2)</th>
<th>Entity Name</th>
<th>Category1</th>
<th>Category2</th>
<th>Category3</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>150</td>
<td>AAA</td>
<td>200.67</td>
<td>130.58</td>
<td>450.00</td>
</tr>
<tr>
<td>200</td>
<td>47</td>
<td>BBB</td>
<td>300.05</td>
<td>50.14</td>
<td>270.88</td>
</tr>
<tr>
<td>130</td>
<td>260</td>
<td>CCC</td>
<td>100.79</td>
<td>158.00</td>
<td>369.88</td>
</tr>
</tbody>
</table>
Because the application system is totally Java-based, it has good compatibility and platform independence. The user can enter the system from any browser supporting Java, either by loading an existing multiple perspectives map application or by creating a new application. When loading, the web server will download the Java Applet code and map image file as well as category data and build a multiple-perspectives map based on these category data. When creating a new application, only the Java Applet code and map image file are downloaded.

At present, the system supports two visual application modes to produce different views based on the data classification in Table 1-1.

The system can be readily modified to support additional modes, however. It currently represents a feasible information visualization solution and design concept for interactive online generation of multiple-perspectives maps. The ease of development of additional visualization application modes based on this solution and design concept is the primary focus of this thesis.

Information Visualization Product and Development

Visualization is an extremely active and burgeoning area of research. In the last few years its development occurred at a rapid pace. One indication is that the 5th International Conference on INFORMATION VISUALISATION took place in London on the 25th-27th of July 2001 [7]. More than 100 papers were presented on world-wide level research about topics like the application of information visualization to construction, medicine, education, parallel and distributed systems programming, geographical information systems and simulation [8]. Another example is that a new, peer-reviewed international journal titled “Information Visualization” was opened in 2002, which is a central forum for all aspects of information visualization and its applications [9].
Visualization products are so extensive and widespread that it is probably impossible to offer a complete summary. A brief description of some of the very popular software produced by the three major developers of visualization products, however, will give some indication of their current range of uses. The three major developers of visualization software for scientific, engineering and technical industries are Silicon Graphics of Mountain View, CA, Advanced Visual Systems (AVS) of Waltham, MA, and IBM.

OpenGL Performer from Silicon Graphics is a powerful and comprehensive visual programming language toolkit for developers creating real-time visual simulation and other professional performance-oriented 3D graphics applications. The toolkit simplifies development of applications used for visual simulation, simulation-based design, virtual reality, interactive entertainment, broadcast video, architectural walk-through, and computer aided design. The latest major release, OpenGL Performer 2.5, is available for both the IRIX and Linux operating systems [10].

AVS Express from AVS is a cross-platform application builder. It enables object-oriented development of rich and highly interactive scientific and technical data visualizations for Digital, HP, IBM, Linux, SGI, Sun and Windows systems and has found particular success in geo-spatial, medical, oil & gas, finite-element analyses (FEA), and computational fluid dynamics (CFD) applications [11].

AVS also creates data visualization software and solutions that serve business needs. OpenViz is such a product. It can be viewed in Web pages, embedded in Office Suite documents or built into powerful runtime applications with highly interactive 2D and 3D graphics interfaces [12].
IBM's Open Data Explorer (OpenDX) is a toolkit for visualization and analysis. It is currently available on Unix workstations from Silicon Graphics, IBM, Sun, Hewlett-Packard, and Digital Equipment. It consists of an extensive set of modules for turning data into graphics. Users create graphics by selecting modules, placing them on a "canvas", and connecting them with "wires." For example, the Isosurface, Streamline, and AutoColor modules perform the standard visualization functions of creating constant-value surfaces, tracing particle paths through velocity fields, and coloring objects based on a data value, respectively.

OpenDX handles 2D and 3D data types. Based on the data type, a module determines which algorithm to use. If a module receives 2D data, it will produce contours; if it receives 3D data, it will produce isosurfaces. Data can be in the form of a regular grids, deformed quadrangles, or triangles.

OpenDX provides a full set of tools for manipulating, transforming, processing, realizing, rendering, and animating data and supports visualization and analysis methods based on points, lines, areas, volumes, images or geometric primitives in any combination [13].

Organization of Chapters

This thesis consists of six chapters. The following chapter discusses the underlying technologies used in OIMPMS. Chapter 3 outlines the function and use of the software. Chapter 4 describes the system design and implementation details. Chapter 5 presents test results, and Chapter 6 gives conclusions.
CHAPTER 2
UNDERLYING TECHNOLOGIES

In this chapter, we will discuss the underlying technologies used in OIMPMS. This discussion will begin with an overview of Java technology. Next a basic structure paradigm of application based on the Java technology will be brought out. Then deeply analyzing four different database access solutions will be given. This discussion will end with a special question solved in OIMPMS: object serialization.

Overview of Java Technology

Up to now, almost all web-based visualization application systems use Java Applets for the client interface, a Java Servlet or Java Application as the web server interface, and Java Database Connectivity (JDBC) to connect with the database. Java technology is also the core technology used in OIMPMS. Java is currently the best platform with which to build Internet applications because a number of its features make this language well suited for use on the World Wide Web.

First, Java Applets are specially designed to use in browsers. An Applet is a Java program designed to be included in an HTML web document. The HTML document contains tags that specify the name of the Java Applet and its Uniform Resource Locator (URL). The URL is the location at which the Applet byte codes reside on the Internet. When an HTML document containing a Java Applet tag is displayed, a Java-enabled browser downloads the Java Applet byte codes from the URL and uses the Java Virtual Machine to execute the codes from within the HTML document. The Java Applet code is written to allow web pages to contain graphical representations or interactive content.
The ability to download an Applet and its close integration with Web browsers makes Java an ideal medium for Web- and Internet-based development.

Second, Java security mechanisms are also important to its superiority as a platform for Internet applications. Note that a Java Applet can be safely downloaded from any system because security mechanisms exist within the Java Virtual Machine to protect against malicious Applet codes. The Java runtime system verifies the Applet byte codes as they are downloaded to ensure that they are valid and that the codes do not violate any of the restrictions placed on Java Applets by the Java Virtual Machine. These restrictions include what operations they can perform, how they access memory, and how they use the Java Virtual Machine. The purpose of these restrictions is to prevent a Java Applet from gaining access to the underlying operating system or to data on the system.

Third, A Servlet is a Java object class running on the web server to extend the server’s capabilities. Java Servlet greatly facilitates building web-based applications.

Fourth, portability is another one important feature in Java. With this feature, same Java application can be run the on lots of different kinds of computers.

Fifth, Java and databases make a powerful combination via Java’s standard extension, JDBC. JDBC creates a programming-level interface for communicating with databases in a uniform manner. Through JDBC, Java programs can execute SQL statements. This allows Java programs to interact with any SQL-compliant database. Since nearly all relational database management systems (DBMSs) support SQL, and because Java itself runs on most platforms, JDBC makes it possible to write a single database application that can run on different platforms and interact with different DBMSs.
Finally, Java is distributed. It has an extensive library of routines for coping easily with communication protocols. For example, support for TCP, UDP, and basic Socket communication is excellent. This makes creating network connections much easier with Java than with any other programming language. Through use of Applets, Servlets, Remote Method Invocations (RMI), etc., applications can open and access objects across the net via URLs with the same ease as accessing a local file system.

Basic Structure Paradigm of Web-Based System

The Java technology makes it possible to build viewing station with no special software installed beyond the operating system and a network browser. Figure 2-1 shows basic structure paradigm of web-based system.

Figure 2-1. Basic Structure Paradigm of Web-Based System

In this paradigm, Java Applet byte codes are transferred through the web server to the client (viewing station). The browser installed on the client starts the Applet codes in its Java Virtual Machine (JVM). Image or other files and data are also transferred through the web server to the client. After starting, the Applet receives user input, responds to this
event, and calls the object methods either locally or remotely (on the web server) to complete the various functions such as loading images, drawing charts, saving data to a database, etc.

**Database Access Solutions**

In theory, the Java Applet and database server can be linked based on the four kinds of database access solutions described below. These solutions can be classified as possessing either a two-tier or a three-tier connection structure [14].

**Two-Tier Connection Structure**

Figure 2-2 shows a two-tier connection structure with a Java Applet and JDBC on the client and the database installed on the web server.

![Two-Tier Connection Structure](image)

Figure 2-2. A Two-Tier Connection Structure of Java Applet and Database

The Applet will connect to the database through JDBC APIs. There is no Java Servlet running on the web server. The two-tier model requires the databases as well as all the files used by applications to be located on the web server that the Applet is loaded from. This requirement is due to Java’s security mechanism restrictions. Because of the resulting lack of flexibility, this structure is only suitable for simple and limited applications. Thus applications are more commonly produced using the three-tier connection structure.
Three-Tier Connection Structure

The three-tier connection structure adds a middle-tier with a Java Servlet or Java Application running on the web server. When an Applet requires any database operation or file operation, it sends an operation message (command) to the Java Servlet or Java Application on the web server. The Java Servlet or Application executes the operation and connects to any server on the Internet as needed to acquire data and to transfer results to the Applets. This makes it possible for the database and files to be distributed among different database servers and file servers on the Internet. Flexibility is thus markedly improved due to this three-tier structure.

There are three alternative modes for the three-tier connection structure with Java Applets on the clients, a Java Servlet or Java Application on the web-server (middle tier), and the database and files on separate database servers and file servers. The database server connection driver is realized in all cases by JDBC. The difference in the three modes occurs in the composition of the middle tier and in the network protocols/services between the client (Java Applet) and middle tier (Java Servlet/Application). The network protocols/services that can be used here are Java’s RMI, socket, and the Hypertext Transfer Protocol (HTTP).

When the middle tier is a Java Servlet embedded in the web server, the default HTTP protocol/service is used. Figure 2-3 depicts this structure. A Java Servlet can be started interactively from the client and run in the web server’s virtual machine. One advantage of using this model is that it is easy to implement and deploy a new service. Another advantage is that the Java Servlet can be accessed easily through the standard port 80, which is accessible through almost any firewall [6]. As security issues become increasingly important, this advantage may become an overriding one.
The disadvantage of this model is that the request-response communication is limited to the parameter string [14]. It is difficult to express a complex request using just the parameter string. In this thesis, a method is presented and the problem is solved successfully. The key to this method is the creation of an object (Message) that extends the Hashtable class. The Hashtable class supports serialization and makes it very easy to compose any complicated information structure. Thus Message objects with complicated message structures can be passed in serialized form. A detailed description of this method is provided in Chapter 4.

With a Java Application running on the web server, RMI protocol/service is a good choice. Figure 2-4 shows the structure using RMI. RMI is Java's remote procedure call (RPC) mechanism [15]. It facilitates object method calls between Java Virtual Machines and dynamically loads new object types as required. When Java Applets communicate by RMI with a Java Application running in advance on the web server, the Java Applets will remotely call the object methods in the Java Application. These methods may realize any
available function, such as connecting to a database, executing a query, transferring a query result, loading an image file, etc. RMI provides a good solution for realizing RPC. It is much easier and simpler to use RMI to develop more flexible services and to build complex applications. The drawback is that many firewalls are configured to limit access to a small number of standard ports (ftp, http, etc.), which means the server cannot be accessed outside of the local firewall [6]. This represents a quite serious limitation in some cases.

Figure 2-4. Three-Tier Connection Structure (RMI) of Java Applet and Database

With a Java Application running on the web server, using socket protocol/service is another alternative. Figure 2-5 shows this kind of structure. A socket can be thought of as an application-level endpoint, described by a port number and an Internet address [16]. When a Java Applet on a browser wants to communicate with a Java Application on the web server, the Applet needs to create a socket using a specific port number and its host Internet address. On the server side, the Java Application must start in a specific socket
in advance, waiting for the Java Applet’s requests. Thus the Java Applet on the browser and Java Application on the server are identified by a pair of sockets. Using user datagram protocol (UDP) or transfer control protocol (TCP), messages are transferred between the two sockets. Sent messages are queued at the sending socket until they have been transmitted. When they arrive, the messages are queued at the receiving socket until the receiving process makes the necessary calls to receive them.

Figure 2-5. Three-Tier Connection Structure (Socket) of Java Applet and Database

Object Serialization

Java is object-oriented, and the capability to store and retrieve Java objects is essential to building almost all Java applications. The key to storing and retrieving objects is representing the state of objects in a serialized form sufficient to reconstruct the objects [17]. The process of converting a set of object instances into a linear stream of bytes is called object serialization. Specifically, in the database access solutions introduced above, object serialization is a key process to pass objects between JVMs.
Very fortunately, object serialization is a mechanism built into the core Java libraries. In order to realize serialization, the object class must implement a java.io.Serializable interface. The basic process of serializing an object is to create an instance of ObjectOutputStream and call the writeObject() method. Reading in a serialized object is achieved by creating an instance of ObjectInputStream and calling the readObject() method. A false code example is provided below to illustrate the process (Figure 2-6).

```java
// for the client side to connect to the server side
URLConnection connection = url.openConnection(); // url is the url of the server side
ObjectOutputStream outputToServlet = new ObjectOutputStream(connection.getOutputStream());
outputToServlet.writeObject(object);
outputToServlet.flush();
outputToServlet.close();

ObjectInputStream inputFromServlet = new ObjectInputStream(connection.getInputStream());
ret = (object) inputFromServlet.readObject();
inputFromServlet.close();
```

Figure 2-6. Object Serialization Process Example Code

A class is not made serializable, however, by just adding the java.io.Serializable interface to its class definition. It is necessary that all the instances and local variables in the class be serializable and that all its superclasses be properly serialized. Java’s primitive data types support serialization. If a class has any local variables or superclasses that are not serializable, it can only be serialized by overwriting the readObject() and writeObject() methods.

Summary

The reader has now been provided a brief introduction to the technology used in this thesis project. This introduction includes an overview of Java technology and of a basic structure paradigm for web-based application systems that will clarify the more technical discussion in the following chapters for those possessing a basic familiarity with computer application systems. A more detailed examination of database access solutions
and object serialization was also included. The emphasis on these latter two topics
coincides with the focus of this thesis project, as they provide the key to the
implementation of the information visualization solution developed therein.
CHAPTER 3
SOFTWARE FUNCTION

This chapter begins with an introduction to the two visualization modes currently available in the system. Then the software functions available in OIMPMS are described in detail. A list and explanation of every available operation is included in this description.

Visualization Modes

Table 1-1 provides the typical data classification support description for OIMPMS. As mentioned in Chapter 1, the system currently provides two visual application modes. The different views provided by these modes enhance the system’s effectiveness for graphical presentation of a broad variety of data types. The user can choose a mode for viewing data appropriate to the data’s category and meaning.

In MODE1, the location coordinates (x, y) correspond to a point on the map and are matched with an entity item representing the geographic area in which that point is located. For example, a point within the Florida state map can represent a city such as Gainesville or Miami or any other geographic region surrounding that point; a point within a map of the University of Florida campus can represent a building or an area within the campus. In this mode, chart locations are distributed among the different points on the map. Every chart depicts a row of data such as that in Table 1-1, with each row representing an entity item. Three kinds of graphical representation (bar, column, and pie charts) are available in this mode. Here the listening area for an entity item is a pie-shaped or rectangular area centered on the point (x, y).
In MODE2, the location coordinates \((y_1, y_2)\) define a rectangular geographic region on the map. The height of the region is \(y_2 - y_1\), and the width of the region is the width of map. All the regions are defined one by one, and there is no gap between any two regions. In this mode, bar charts are drawn on the right side of the map next to the defined region. The vertical dimension of a bar is the same as that of the defined region \((y_2 - y_1)\), while its length is determined by the category data associated with the defined region. Here the listening area for an entity item is the region itself.

Figure 3-1. Loading an Application

**Software Functions**

**Loading an Application**

When the LOAD button is clicked, a LOADING frame is displayed and all the existing visualization applications in the system are presented to the user by the ComboBox component (Figure 3-1 above). Selecting the desired application and clicking the LOAD button within this frame will cause the system to load the map and all the data
related to the map, drawing the charts in or beside the map based on the data and mode. If there is no application in the system, a dialog box with the prompt "No Application Can Be Loaded!" will be displayed. The CANCEL button will cancel the loading operation.

If an application exists in memory, a loading operation will display a dialog box, allowing the user to save or not to save the current application.

Creating a New Application

When the NEW button is clicked, a process to create a new multiple-perspectives map application will begin. If an application exists in memory, an operation creating a new application will first display a dialog box that allows the user to save or not to save the current application. In creating the new application, an empty SETUP frame is displayed first, in which the user can input the SETUP information for the new application. The SETUP information is the general profile information for an application, including the application name and description along with values for VisualMode, TotalCategory, CategoryName, ImageName, ChartType, etc. The SETUP frame is depicted below (Figure 3-2).

"VisualMode" stores the user’s visualization mode choice, while "ChartType" stores the user’s choice of chart type. If Mode2 is selected, this field will be disabled because only bar charts are available in Mode2. The "BasedSize" field controls the size of every chart on the map, so changing the value of this field will cause the size of every chart on the map to change. A more detailed explanation of this feature in the Chapter 4.

"TotalCategory" is the total number of data categories in an application. It must be a number from 1 to 10 because the system can process a maximum of ten data categories. After a number is input to the "TotalCategory" field, the system will automatically assign the default values for category name to the following fields. For example, if the user
inputs 3 to the “TotalCategory” field, the system will automatically assign CATEGORY0 to the “C0” field, CATEGORY1 to the “C1” field, and CATEGORY2 to the “C2” field. Users can change these category names if they prefer. Once the OK button is clicked, the system will check the input SETUP information. If any error is found, an error prompt dialog box will be displayed, and the system will return to the SETUP frame. If all the input information is correct, an image map with the name defined in the SETUP frame will be loaded. All the input SETUP information will be stored in memory until a SAVE operation. Next, the user can proceed to other functions.

![Setup an Application](image.png)

**Figure 3-2. Application Setup**
Modifying an Application Setup

When the SETUP button is clicked and if an application has already been loaded or created, a SETUP frame filled with the general profile information for the current application will be displayed (Figure 3-2 below). All the setup information can be changed in this frame except the image name and visual mode. Once the OK button in the SETUP frame is clicked, the current application will be modified and adjusted according to the SETUP changes entered. For example, if a new category is added, a default category name CATEGORYn will be assigned to “Cn” field unless the user also changes the value in this field. Every entity item will be assigned a value of 0 for this new data category. If a category is deleted, the values assigned for that data category in every entity item will also be deleted. Clicking the Cancel button will cancel any changes that the user has entered, and the original SETUP information will be retained.

If no application is loaded in the system, a setup operation will also cause a message dialog box to be displayed, informing the user that no setup operation is available.

Saving an Application

When the SAVE button is clicked, a confirmation dialog box is displayed. Once the user confirms the request to save the application, an event is activated to save the currently loaded or created application to the databases, including its SETUP information and all the entity items data. If the operation is successful, a dialog box with the message “Saved Successfully” is displayed.

Deleting an Application

When the DELETE button is clicked, a confirmation dialog box is displayed. Once the user confirms the request to delete the application, an event is activated to delete the currently loaded or created application from the databases and memory. If the operation
is successful, a dialog box with the message “Deleted Successfully” will be displayed. After the delete operation, the system will wait for the user to load or create another application.

**Adding an Entity Item**

Moving the mouse pointer over the map image and clicking at any location outside of the existing entity item listener areas will activate an event to add an entity item. The event action will display a frame titled “Add An Entity Item” (in MODE1) or “Add an entity region” (in MODE2) and wait for the user to input the entity name and the category data values (Figure 3-3).

![Add An Entity Item](image)

**Figure 3-3. Adding an Entity Item**

In MODE1, the coordinates of the clicked point are shown in the “X Coordinate” and “Y Coordinate” fields of the displayed frame. In MODE2, a line is drawn across the width of the map at the y-axis coordinate at which the mouse click occurred. This y-axis coordinate becomes the lower boundary (y2) for the new entity item region, while its upper boundary (y1) coincides with the lower boundary of the entity item region immediately above it. The y-axis coordinates for the upper and lower boundaries of the
new entity item region are shown in the “Y Beginning” and “Y Ending” fields of the displayed frame.

All categories take data of type float. If a character other than a number or dot is input, the system will block this input and indicate with a beep that an inappropriate character key has been pressed. Once the OK button is clicked, the new entity item will be added, and a new chart representing this new item will be displayed on the map. All the input data is kept in memory until the SAVE button is clicked. Clicking the Cancel button will cancel the action of adding an entity item.

Figure 3-4. Inspecting and Modifying an Entity Item
Inspecting, Deleting, and Modifying Entity Items

Clicking the mouse within an existing entity item listener area activates an event to select the entity item and allow the user to modify its properties. A blue dotted line is drawn on the map to indicate the current borders of the listener area for the selected entity item. A frame with an enlarged entity item chart along with the entity item data is then displayed (Figure 3-4 above). There are three buttons in this frame in MODE1. Clicking the DELETE button will cause the selected entity item data to be deleted from memory and the corresponding chart to be immediately removed from the map. Alternatively, the user can modify any entity item data value and confirm this change by clicking the MODIFY button. Upon confirmation the modified data will be stored in memory, and the corresponding chart on the map will immediately be modified to reflect the change in the data. The CLOSE button closes the frame without any further action. In MODE2, an INSERT button will also be available.

Moving an Entity Item (MODE1)

When the mouse pointer is within an existing entity item listener area, pressing a mouse button and moving the mouse pointer will drag the entity item chart to the location on the map at which the mouse button is released. The values in the “X Coordinate” and “Y Coordinate” fields of the corresponding entity item will change accordingly. This function is limited to Mode1 applications.

Inserting an Entity Item (MODE2)

As previously mentioned, the user in MODE2 defines multiple adjoining rectangular regions sequentially. Each of these rectangular regions is an entity item. A new rectangular region and corresponding entity item can be inserted within any of the existing rectangular regions. When inserting an entity item within an existing region, the
“Y Ending” field for the existing region must be redefined. The entity item region in which the insertion occurs is split into two adjoining regions by this operation.

Dragg a and Moving a Borderline (MODE2)

The bottom borderline of a defined region can be dragged in either vertical direction. This motion will change the height of the defined region. The change will be reflected in the memory data immediately, and charts on the map will be changed accordingly.

Summary

This chapter describes the functions available to the user in the OIMPMS system. Detailed instructions for the use of these functions are presented from the perspective of a typical user. Through this description, the user can obtain a general overview of the system along with the more specific directions required for the effective use of the functions it provides.
CHAPTER 4
SYSTEM DESIGN AND IMPLEMENTATION

This chapter will discuss the detailed system design and implementation. The description of the system design will emphasize the design thought and software structure design. The description of the system implementation will focus on the environmental setup and software installation details.

System Design

The discussion in Chapter 2 indicated why Java is the best platform for Web applications. OIMPMS is designed to use Java technology and is written completely in Java. Furthermore, the three-tier database access structure using HTTP protocol was chosen because of its ease of implementation and its accessibility through almost any firewall. Finally, an effective method for allowing any complicated message to be passed between an Applet and Servlet was developed. Thus the problem caused by the typical limitation of the structure is successfully solved. The system design will be described in five parts: Java Applet, communication link, database design, Java Servlet and objects design. Object serialization will be discussed in relation to the communication link and objects design.-

Java Applet

The keys to the Applet design are the interface layout and event processing ability. A user-friendly interface is undoubtedly an important aspect of a successful visualization system, while powerful event processing ability is a requirement of the dynamic and interactive features of a visualization system. All the Applet codes are put in a file
(getdata.java). This file is composed of getdata, an object that is an extension of the JApplet class, and of many other objects extending the JInternalFrame and JPanel classes. The layout managers GridBagLayout, BorderLayout, GridLayout, and FlowLayout are used to build the interface easily and in a user-friendly way. The interfaces ActionListener, MouseListener, MouseMotionListener, and FocusListener are implemented to catch ActionEvent, MouseEvent, and FocusEvent, respectively, when an event occurs.

All the components are Java Swing components in the Applet code. Compared to AWT, Swing is platform-independent and has richer graphical user interfaces and more powerful components. Event processing in Swing is also more flexible.

Communication Link

Building the communication link between Java Applet and Java Servlet is a necessary step for OIMPMS. An object (servlet_runner) is created and the method connectServlet (URL, Message) is defined in this object to build this communication link. Parameter URL is a URL object instance of Servlet’s Uniform Resource Locator. Calling the URL object method openConnection () will return an URLConnection object instance that represents a communication link between the Applet and Servlet. The other parameter, Message, is a Message object instance. We mentioned in Chapter 2 that Message is an object extending the Hashtable class to build a complicated request. Applet object getdata calls the method connectServlet (URL, Message) by giving Servlet’s URL and a Message object instance to build a link to Servlet and pass the Message object instance to Servlet. A critical code paragraph of the method connectServlet (URL, Message) is given in Figure 2-6.
The aim of creating the Message object is to take full advantage of the Java object serialization mechanism and pass complicated messages in serialized form. In the system, the messages could be any combination of queries, many instances of objects, the type of operation, etc. In the discussion in Chapter 2, we indicated that the object serialization mechanism makes it possible to pass serialized objects between JVMs. Message object is a subclass of Hashtable class and implements a java.io.Serializable interface. The support for serialization by the Hashtable class is an invaluable feature, allowing us to construct the passed messages by mapping the queries, objects instances, etc. to keys in the Message object. The following is a false code example of the construction of a Message object instance.

```java
Message Msg;
Msg.put("0", operationtype);
Msg.put("1", Querystring);
Msg.put("2", numberofobjectinstance);
Msg.put("3", firstofobjectinstance);
Msg.put("4", secondofobjectinstance);
...
Msg.put("n", ...);
```

The Message object makes it possible to compose any complicated messages. Different Message object instances will result in different function realizations.

**Database Design**

Two tables (V_SYSTEM1 and V_SYSTEM2) are created according to the system’s requirements for a given application. The structure of these tables is presented in Figure 4-1. V_SYSTEM1 stores the profile information of every multiple-perspectives map application. “ID” is the primary key of V_SYSTEM1. Ten fields (“C0” to “C9”) are created to store the names of data category items. “TOTALCATEGORY” is created to save the total number of data category items. A maximum of ten data categories can be processed by the system. If a multiple-perspectives map application only includes five
categories, “C5” to “C9” will be assigned null value. “MAXDIAMETER” is created to
determine the chart size of the entity point possessing the maximum category data sum.
By changing the value, users can change the size of all the charts of associated with the
entity points on the map. A detailed discussion of this feature will be included in the
object design section. “CHARTTYPE” is created to allow the choice of chart type among
pie, bar, and column. “VISUALMODE” is created to store the visualization mode.

<table>
<thead>
<tr>
<th>V_SYSTEM1</th>
<th>V_SYSTEM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID &lt;pk&gt;</td>
<td>ID &lt;pk&gt;</td>
</tr>
<tr>
<td>IMAGENAME</td>
<td>IMAGE</td>
</tr>
<tr>
<td>TOTALCATEGORY</td>
<td>TOTALCATEGORY</td>
</tr>
<tr>
<td>C0</td>
<td>C0</td>
</tr>
<tr>
<td>C1</td>
<td>C1</td>
</tr>
<tr>
<td>C2</td>
<td>C2</td>
</tr>
<tr>
<td>C3</td>
<td>C3</td>
</tr>
<tr>
<td>C4</td>
<td>C4</td>
</tr>
<tr>
<td>C5</td>
<td>C5</td>
</tr>
<tr>
<td>C6</td>
<td>C6</td>
</tr>
<tr>
<td>C7</td>
<td>C7</td>
</tr>
<tr>
<td>C8</td>
<td>C8</td>
</tr>
<tr>
<td>C9</td>
<td>C9</td>
</tr>
<tr>
<td>MAXDIAMETER</td>
<td>MAXDIAMETER</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>NAME</td>
<td>NAME</td>
</tr>
<tr>
<td>CHARTTYPE</td>
<td>CHARTTYPE</td>
</tr>
<tr>
<td></td>
<td>&lt;pk,fk&gt;</td>
</tr>
</tbody>
</table>

Figure 4-1. Database Table Structure

V_SYSTEM2 stores the category data and map coordinates for every entity in an
application. Fields “X” and “Y” are created to store the coordinate on the map at which a
chart is centered; thus they correspond to an entity in MODE1. In MODE2, fields “X”
and “Y” are used to store the y coordinate of the top and bottom borderline. Ten fields
“C0” to “C9” are created to save the category data values for an entity. Again, if a
multiple-perspectives map application only has five categories, “C5” to “C9” will be
assigned null value. “ENTITY” is created to save the entity name. “BELONGTO” is the
foreign key of V_SYSTEM2 and is related to the primary key “ID” of V_SYSTEM1.
Java Servlet

The Servlet parses the received Message object instance, executes the requested function, and sends the result back to the Applet. The function modules realized by the Servlet are INITIALIZE, LOAD, SAVE and DELETE. Their function descriptions are as follows.

INITIALIZE: load all the records in the V_SYSTEM1 to memory.

LOAD: load the records of all the entity points of the loaded multiple-perspectives map application from V_SYSTEM2.

SAVE: save the currently loaded multiple-perspectives map application to the database.

DELETE: delete the currently loaded multiple-perspectives map application from the database.

The only Servlet object class in the system is receive_applet_servlet. In the communication link section of this chapter, we saw that the Applet object getdata makes this object run by providing its URL and an instance of Object Message. The Servlet object receive_applet_servlet fetches the Message object instance and retrieves its values by the appropriate keys and finishes the correlated function module and returns a Message object instance composed of the results to getdata.

JDBC API is realized in methods login () and logout () in a separate object (db). The Servlet calls methods login () and logout () to open and close the database for every realization of a function module.

The realization of all the function modules goes through the following steps:

1. Analyze the passed messages to decide which database to open and what operation to execute;
2. Call db.login() to open the database;
3. Execute the database operation;
4. Call db.logout() to close the database;
5. Return the result (Message object instance).

Object Design

Objects SetupData and chartdata are designed to represent the two database tables. They all implement a java.io.Serializable interface to be passed in the Message object. All the local variables in SetupData and chartdata are serializable variables. There is no any other work to perform to realize serialization.

Objects column, bar, and pie are designed to draw the charts. The codes were written in Swing. Here one point needs to be clarified: the chart area size in MODE1. For pie chart, the chart area is a circular area. For bar and column charts, the chart area is rectangular. We designed the multiple-perspectives map application so that the size of the chart area has a meaning: the category data sum. If the category data sum of an entity point is larger, then the chart area is larger and the chart is therefore larger. We mentioned in the database design section that “MAXDIAMETER” sets the size of the chart area for the entity point that has the largest data sum. The size of the chart area for any other entity point is then proportional to that of the entity point with the largest data sum, based on the ratio of its data sum to the largest sum. Changing the value of “MAXDIAMETER” will change the size of all the charts on the map. In MODE2, the situation is much simpler. The category data determine the height of the bar charts. The method used for this determination is the same as that used to set the chart sizes in MODE1.
The bar, column, and pie object also includes methods to draw the charts in a separate panel. The size of these charts is determined by the panel size. The methods for adding labels, gridlines, coordinate axes, data values, etc. are also included in these objects. These methods allow the application to enlarge the chart for an entity to any size.

Finally, objects JFloatField and JIntegerField were created to ensure that numbers of type float and integer, respectively, are input as required by the fields in the database.

**System Implementation**

**System Environment Setup**

Netscape in the Unix and Windows environments and Internet Explorer in the Windows environment will be used as Web browsers. Apache 1.3.19 will be used as the Web server and Tomcat as the Servlet server. Tomcat is a Servlet container with a JSP environment. A Servlet container is a runtime shell that manages and invokes Servlets on behalf of users. Sybase will be used as the back-end RDBMS.

Table 4-1. Tomcat directory structure table.

<table>
<thead>
<tr>
<th>Directory Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bin</td>
<td>Contains startup/shutdown... scripts</td>
</tr>
<tr>
<td>Conf</td>
<td>Contains various configuration files including server.xml (Tomcat's main configuration file) and web.xml, which sets the default values for the various web applications deployed in Tomcat.</td>
</tr>
<tr>
<td>Doc</td>
<td>Contains miscellaneous documents regarding Tomcat.</td>
</tr>
<tr>
<td>Lib</td>
<td>Contains various jar files that are used by Tomcat. On UNIX any file in this directory is appended to Tomcat's classpath.</td>
</tr>
<tr>
<td>Logs</td>
<td>Contains Tomcat’s log files.</td>
</tr>
<tr>
<td>webapps</td>
<td>Contains sample web applications.</td>
</tr>
</tbody>
</table>
After installation, Tomcat should have the main directory structure given in Table 4-1.

The configuration file “server.xml” must be modified to be suitable for the specific application. An excerpt from “server.xml” is listed in Figure 4-2.

```
<ContextManager home="/cise/homes/jhuo/ThesisProject" debug="0" workDir="work" >
   ...
   <Connector className="org.apache.tomcat.service.SimpleTcpConnector">
      <Parameter name="handler" value="org.apache.tomcat.service.http.HttpConnectionHandler"/>
      <Parameter name="port" value="51639"/>
   </Connector>
   ...
   <! -- project servlet and jsp code directory -->
   <Context path="/project" docBase="Webapps/proj" debug="0" reloadable="true" >
      ...
   </Context>
</ContextManager>
```

Figure 4-2. An Excerpt From “server.xml” File

Software Installation

After the system environment is setup, we installed the software in the webapps/thesis directory. The code directory structure is as follows:

thesis:

- Applet: getdata.java, servlet_runner.java, thesis.jsp
- jsp: pie.java, column.java, bar.java, SetupData.data, chartdata.java, Message.java, JFloatField.java, JIntegerField.java,
- WEB-INF:
  - classes: receive_applet_class.java
  - jsp: chartdata.java, Message.java, db.java, SetupData.java
- IMAGES:
  - MAPS:

Summary

This chapter provides a detailed description of the system design. Every component and every problem solution in the design is explained. Through this explanation, we seek
to provide a clear description of the design thought and solution. In the final section, some important aspects of the implementation are mentioned.
CHAPTER 5
TESTS AND RESULTS

In this chapter, state-based testing in object-oriented programming is discussed first and system state diagrams of OIMPMS are presented within the context of this discussion. The two sample applications used to implement state-based testing of the system functions in accordance with the system state diagrams are discussed next. Visualization result figures derived from the two sample applications are also given and briefly described.

State-Based Testing

Testing is an important part of the software development process, providing a systematic means for detecting bugs, improving software design, and reducing usability problems. There are a number of test techniques available, such as white box testing, black box testing, state-based testing, system testing, and integration testing.

Object-oriented systems build upon the notion of classes, which link sets of objects to form larger systems such as packages or applications. Testing an object-oriented system may be done at the individual object level, but it will more commonly involve accessing a set of objects through a small number of methods operating on a small number of objects. Tested components can be treated as white or black boxes. In black box testing, knowledge of internal implementation structures is lacking or ignored, while white box testing depends on such internal knowledge to drive the tests [18].

In object-oriented programming languages, state-based testing derives test cases by modeling a class as a state machine. Methods result in state transitions, and the state
model defines allowable transition sequences. Test cases are devised to exercise each transition [19]. Because black box testing of an object is essentially limited to the object’s methods as tested components, state-based testing of an object-oriented system encompasses black box testing as well.

We will focus on state-based testing in this discussion.

![System State Diagrams](image)

**Figure 5-1. System State Diagrams**

In OIMPMS, there are two states: Initialization and Application Running. The methods resulting in state transitions are: LOAD, NEW, SETUP, SAVE and DELETE.
When the system is in the Initialization state, only the LOAD and NEW methods can be executed. When the system is in the Application Running state, all the methods are available.

Figure 5-1 provides the system state diagrams, which clearly describes the state transitions of the system. In the diagrams, a method name placed above a line with arrow represents a button clicked by a user; a method name splitting a line with arrow represents a method executed by the system. “Yes,” “No,” “Cancel NEW,” and “Cancel LOAD” are input choices available in a confirmation or option box. Testing of the system was designed to execute every path depicted in the state diagrams.

Test Implementations and Results

Two sample applications were to target the two visualization modes in the system. Through the creating, loading, setup, saving, and deleting of the two sample applications and the performance of data input operations, every path in Figure 5-1 can be executed. Thus every object and method will be examined.

Table 5-1. Sample of the Test Data used in Example 1.

<table>
<thead>
<tr>
<th>State Name</th>
<th>1996</th>
<th>1997</th>
<th>1998</th>
<th>1999</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alabama</td>
<td>20,329</td>
<td>21,129</td>
<td>22,118</td>
<td>22,972</td>
</tr>
<tr>
<td>Alaska</td>
<td>26,057</td>
<td>27,080</td>
<td>27,950</td>
<td>28,629</td>
</tr>
<tr>
<td>Arizona</td>
<td>21,611</td>
<td>22,780</td>
<td>24,133</td>
<td>25,173</td>
</tr>
<tr>
<td>Arkansas</td>
<td>19,442</td>
<td>20,228</td>
<td>21,256</td>
<td>22,233</td>
</tr>
<tr>
<td>California</td>
<td>25,563</td>
<td>26,742</td>
<td>28,264</td>
<td>29,856</td>
</tr>
<tr>
<td>Colorado</td>
<td>26,231</td>
<td>27,951</td>
<td>29,856</td>
<td>31,533</td>
</tr>
<tr>
<td>Connecticut</td>
<td>33,472</td>
<td>35,619</td>
<td>37,861</td>
<td>39,543</td>
</tr>
</tbody>
</table>

The first example is the “United States Per Capita Personal Income by State” visualization application. This application uses MODE1. There are four data categories: 1996, 1997, 1998 and 1999. They represent per capita personal income (in dollars) for the
specified years. This application is based on a United States map. The data comes from the United States Bureau of Economic Analysis. Table 5-1 above gives a sample of the data.

Figure 5-2. Visualization Result – Pie Chart of Example 1

Figure 5-2 gives a visualization result from the application used in Example 1. A separate pie chart is displayed for each state to depict its per capita personal income data. The color of a slice indicates the year to which the data value applies: red for 1996, blue for 1997, yellow for 1998, and green for 1999. The chart’s BasedSize is 20.0 pixels. As indicated in the discussion in Chapter 4, the size of the chart is determined by the sum of its category data. By comparing the sizes of the charts, we can obtain a general knowledge of the distribution of poor and rich states within the United States. Tool tips in
the figure are used to show the chart size (in pixels) when the mouse enters the listening area associated with it. This helps the user to accurately compare the sizes of the respective charts.

Figure 5-3 provides an alternative visualization result, also from Example 1, obtained by changing the chart type to column chart. The four columns distributed along the x-axis from left to right represent data from 1996 to 1999. Besides the visualization meanings provided in Figure 5-2, the column charts also indicate clearly to the user that income was increasing steadily from 1996 to 1999 in every state in the United States.

The second simple example was created using virtual data and a map of Florida. Only one category exists in this application, which uses MODE2. This application divided the
State of Florida into 6 rectangular regions from north to south. For every region, we assumed that a statistical data value had been derived from an economic index of interest.

Figure 5-4 gives a visualization result from Example 2. As can be seen in this figure, the vertical dimensions of the bars in the bar chart are identical to the vertical dimensions of the corresponding rectangular regions, and the lengths of the bars reflect the data values derived from the economic index.

Figure 5-4. Visualization Result of Example 2

Summary

In this chapter, the results of the state-based testing of the system are presented. Two examples used in implementing the tests are described, and figures depicting the visualization results from the tests are provided. These figures demonstrate the achievement of the intended functionality by the system, in which all functions perform as expected.
CHAPTER 6
CONCLUSIONS

This chapter provides a summary of OIMPMS emphasizing its most noteworthy features, which are described in detail in the preceding chapters. Some suggestions for future improvement of the system are also offered.

Summary of OIMPMS

OIMPMS is an information visualization application system featuring online, interactive generation of multiple-perspectives maps. Any Internet user can use the system to create, modify, save, and delete multiple-perspectives map applications with no special software installed beyond an operating system and a network browser. A multiple-perspectives map application supports visualized presentation of charts derived from category data (such as population projections by the Census Bureau). The charts are distributed on a map according to the geographic locations corresponding to the underlying data. Thus in addition to providing the viewer with a visual representation of the data’s information contents for each location individually, a multiple-perspectives map also presents the geographic characteristics of the data in an accessible visual form.

The system is interactive in two respects: one corresponds to the capacity for interface-friendly visualized input of data for the user-specified category and the other is its support for selection by the user from among multiple visualization application modes and chart types for inclusion on the maps. This latter feature allows a multiple-perspectives map of specific mode and specific chart type to be generated. Currently OIMPMS supports three chart types and two application modes for building multiple
perspectives maps that differ with respect to chart positioning and geographic region
demarcation.

Some Suggestions for Improving OIMPMS

Although OIMPMS has many interactive features, there are some improvements that
can be made about the interactivity of the system.

First, Allow the user to browse some or all of the local disk directories during Setup
and select a map image file from one of these directories. The purpose would be to
upload the selected map image file to the directory designated to store the map image
files on the web server or other file server. Thus the user would be able to select a map
image file more flexibly.

Second, Provide a powerful user-friendly interface for designating the data source of
the categories in an application. Currently the only source for data is user input. But we
often encounter a situation in which two applications share one or more data categories
possessing identical data values. For example, Application A might be “United States Per
Capita Personal Income by State Visualization Application (1996-1999)” and Application
B “United States Per Capita Personal Income by State Visualization Application (1998-
2002).” These applications share two categories (“1998” and “1999”) with identical data
values. If we have already created Application A and input all its data and now want to
create Application B, we should be provided a way to designate the data source for
categories “1998” and “1999” in Application B as the corresponding categories in the
database associated with Application A. Thus the interface would permit the data sources
for the categories in an application to include other applications as well as user input.
Summary

OIMPMS represents a feasible information visualization solution and design concept for interactive online generation of multiple-perspectives maps. Based on this solution and design concept, additional visualization application modes can easily be developed. Moreover, the usability and scope of the visualization applications offered by the system will be greatly enhanced with further improvements in its interactivity along with the development of additional chart objects.
REFERENCES


BIOGRAPHICAL SKETCH

Jing Huo was born in Taiyuan, Shanxi, P.R.China. She received her Bachelor of Engineering degree in computer science and application from Southwest Jiao Tong University, P.R.China, in 1991. She worked as a network administrator in the National People’s Congress, Beijing, from 1991 to 2000. In the fall of 2000, she began her graduate study in the Computer and Information Science and Engineering Department of University of Florida. Her interests include Web applications, database applications and other software development. She is also interested in system and network administration. She will receive her Master of Science degree in computer engineering through the College of Engineering in August 2002.