APPLYING 3D VISUALIZATION TECHNIQUES

TO THE REPRESENTATION OF

SOFTWARE ARCHITECTURE

THESIS

Presented to the Graduate Council
of Texas State University-San Marcos
in Partial Fulfillment
of the Requirements

for the Degree

Master of SCIENCE

by

Tsz Muk (Jimmy) Kung

San Marcos, Texas
May 2006

ACKNOWLEDGEMENTS

I am very thankful to Dr. Hall, Dr. Hazlewood, Mr. Davis and Mr. Grechanik for their guidance and support. Doing a thesis under their guidance has been the most valuable and rewarding experience of my student life. I would like to thank Dr. Chen, Dr. McCabe and Dr. Kaikhah for allowing me to use their students to participate in the survey for this thesis. In addition, I would like to thank Mrs. Trish Sumbera for her support and for providing various lab resources and facilities during my thesis project.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

APPLYING 3D VISUALIZATION TECHNQUES

TO THE REPRESENTATION OF

SOFTWARE ARCHITECTURE

by

Tsz Muk (Jimmy) Kung

Texas State University-San Marcos

May 2006

SUPERVISING PROFESSOR: GREGORY HALL

Software visualization is the mapping from software to graphical representations. Software visualization is needed because software is inherently invisible. Most people visualize source code textually. However, nowadays most software system are designed and maintained in a large scale. Without a good understanding of the software itself, it is very hard to modify, update and debug certain parts of the software system.

This thesis investigates the use of various 3D visualization techniques to represent the large quantities of numerical data needed to describe a large component-based software system. The thesis also examines and adapts existing visualization tools to design a 3D visualization tool to allow a person to easily understand and analyze large and complicated software systems.

CHAPTER 1

INTRODUCTION

When software becomes larger and more complex, the volume of information to be comprehended by the developer or programmer becomes overwhelming. Software visualization has long been used to illustrate information about programs and software. It is only recently that visualization has been designed and implemented in three dimensions. Structures and layouts other than graphs in a three dimensional space are still relatively new concepts [13].

Most software systems are composed of numerous components. An individual component carries a set of functionalities that helps the software to perform different tasks. Well-developed software requires good coordination among the components with each component serving its purpose. Software engineers frequently use a set of software metrics to measure the quality of these components and their relation to each other.

The goals of this project are to address the problems of software visualization, discuss the benefit of using 3D visualization as a software engineering tool, present and critique commercially available visualization tools, and develop a tool that helps a person to understand the basic architecture of a software product and perform effective software analysis.

In this project we developed a 3D visualization tool that combines a couple of effective software visualization techniques such as the object-oriented paradigm, software metrics measurement, and 3D visualization. Our tool collects software metrics from the modules of a given software system. Then the tool transforms the software modules into 3D objects and transforms the software metrics for each module into visual attributes such as size, color and location. Since our research has shown that existing visualization tools do not offer adequate flexibility of views, we provide a mechanism in our 3D visualization tool that allows a user to generate customizable views to further enhance their ability to understand the given software.

Our 3D visualization tool consists of three parts: a back-end engine that serves as a data collector from a given system, a converter that normalizes the data gathered from the back-end engine and formulates them into a hierarchy of objects, and a front-end graphical engine that parses these data and represents them as three dimensional objects with a customizable view. We will discuss this in further detail in chapter three.

In addition to implementing the visualization software, we also set up a survey to test the effectiveness of our visualization tool. We selected a pool of university students who were majoring in Computer Science as our sample space. Then we separated them into groups to participate in two survey tests. The results are analyzed by using a statistical approach to determine if a person is able to accurately and efficiently analyze a given software system with the use of 3D visualization.

The survey result shows us that 3D software visualization indeed makes a significant difference in software analysis. However, based on certain hindering factors, we are not able to successfully determine the true effectiveness of our visualization tool. Though, by carefully examining the results we concluded that, with more careful

planning and the use of repeated testing, we expect to be able to tell the true effectiveness of our visualization tool. The detail of the survey results are described in chapter four.

1.1. The Definition of Software Visualization

Visualization is defined as "the formation of mental visual images; the act or process of interpreting, in visual terms, or of putting into visible form" [Webster, 23]. However, this definition is very broad and not specifically related to any particular discipline. In this thesis, we define software visualization as a mapping from software programs to graphical representations. This means we use analytical methods to extract quantitative data from a system and then turn these data into graphical images that show constructive aspects of the software system so that the user is able to obtain a better understanding of the software system.

1.2. The Dimensions of Software Visualization

It is important for us to address the dimensions of software visualization described by Marcus, Feng and Maletic (2003) in "3D Representation for Software Visualization":

Tasks – *why* is the visualization needed?

Audience – *who* will use the visualization?

Target – *what* is the data source to represent?

Representation – *how* to represent it?

Medium – *where* to represent the visualization? [14]

Failure to properly identify these dimensions while designing a software visualization tool can hinder users' understanding of the software or, even worse, mislead the user to a false or biased conclusion. Throughout the development of this project, we will be sure to address each of these dimensions.

1.3. The Importance of Software Visualization

When approaching unfamiliar software for the first time, even if the software is trivial, most people find it difficult to comprehend. Understanding software is a major problem in industry and research. It costs millions of dollars each year to employ people to analyze software for modification, alteration and correction [12], though, with the right kind of software visualization tool, an interpreter is able to rapidly grasp the selective information provided by the tool and communicate with the software or system. Hence, software visualization reduces time and resources on software development and maintenance.

1.4. Organization of the Thesis

The remaining chapters of this thesis are categorized as the following:

Chapter 2 discusses other research on software visualization, and issues to consider when designing a software visualization tool. Various software visualization schemas and tools are listed and existing software visualization tools are compared to the visualization tool developed in this thesis.

Chapter 3 describes the visualization tool, including the overall architecture, design and implementation, and available features.

Chapter 4 describes the methodology used to test the effectiveness of our visualization tool. It explains the requirements for the survey and describes how we selected our test subjects. This chapter also lists the requirements needed to conduct the experiment and provides instructions for the survey participants. Then it explains the procedure of categorizing our test subjects into groups for different tests in order to reduce bias in the survey. Finally, we make analysis upon the survey result using statistical method.

Chapter 5 presents conclusions and suggestions on future improvements that can enhance the effectiveness of our visualization tool.

CHAPTER 2

LITERATURE SURVEY

2.1. Software can't be Visualized

In F. Brooks's paper "No Silver Bullet" [4] he wrote

"Software is invisible and unvisualizable [,]" and *"...Software is very difficult to visualize. Whether one diagrams control flow, variable-scope nesting, variable cross-references, dataflow, hierarchical data structures or whatever, one feels only one dimension of the intricately interlocked software elephant. If one superimposes all the diagrams generated by the many relevant views, it is difficult to extract any global overview."*

Here is an example which supports Brooks's statement, an entity-relationship diagram of a small well-maintained commercial system (See Figure 2.1). Then a 2D software visualization method is used to display its software architecture. The excessively crowded information makes it really hard for anybody to understand the system without extensive studying.

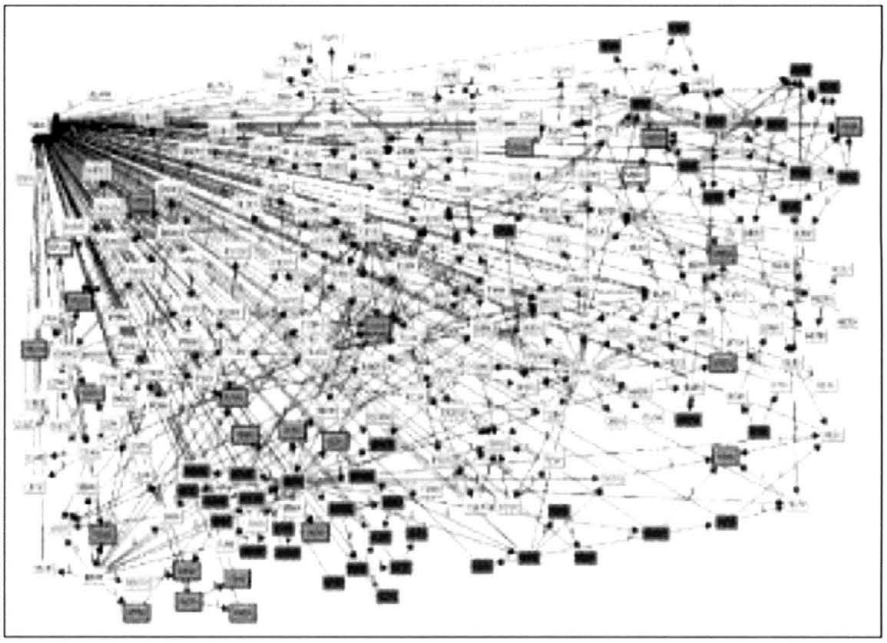

Figure 2.1

So are we foolish for trying to use software visualization to aid software analysis if software visualization is not possible? In *Visualizing Software – A Key Research Area*, Knight and Munro state that Brooks's claim was valid given the resources for hardware and graphics technology that were available twenty years ago. However, with the advancement of 3D visualization, it is likely that well designed software visualization can still effectively overcome the problems of extracting meaningful information from software [13].

In addition, Knight and Munro claim that combining 3D visualization with the techniques of intelligence amplification and visualization metaphor can make effective software visualization possible [13].

Intelligence amplification is the use of computers to enhance the user's understanding and comprehension, as opposed to the artificial intelligence view of trying to replace the human: In this case, visualization as a process by displaying graphically a large and complex data set to try and aid the user's understanding and comprehension, which acts as an intelligence amplification tool [13].

A visualization metaphor is a mapping between a data item and a graphical representation of that data item. A good use of a metaphor that creates a logical framework can further aid users' understanding in software visualization [13]. An example of a visualization metaphor can be shown in Figure 2.1 whereas the rectangles in the pictures are represented as the processes of the system and the lines are represented as the connections between the processes of the system.

In this project we use a 3D visualization tool that acts as the intelligence amplification to aid the user while analyzing software. By definition, intelligence amplification should not replace the human. Hence, our visualization tool allows the user to compose customizable views based on whichever setting the user feels will be most helpful. The engine also allows the user to navigate the view with 360 degrees of freedom, which further enhances the user's ability to examine software system.

Within the view of our visualization engine, we apply the mapping of a set of software metrics for a given software system to 3D objects as a visualization metaphor. The graphical representation of the software data consists of spheres and 3D planes. These spheres and planes have specified dimensions, locations and color settings depending on the user's definitions.

2.2. Existing Software Visualization Schemas

Large, complex software systems are very difficult to understand. Different techniques and approaches have been developed to aid users. Sometimes it helps a person to understand large programs by reducing the amount of detail and using high level abstraction [18]. However, simply providing different levels of abstraction might not be sufficient since users might still be dealing with a large amount of information and data [18]. In addition, "Not every visualization technique is equally usable in displaying a particular dataset. The visualization technique might lack an appropriate navigation support or may not allow the effective reduction of the amount of information displayed through a choice of distinct views" [18]. By discussing and using some of the existing effective software visualization schemas, we have developed a successful software visualization tool.

2.2.1. 2D versus 3D Visualization

"Two-dimensional software visualization is a natural and traditional approach which is still in use today." A good example would be UML – Unified Modeling Language, which is considered as a graph-based language. UML consists of syntax graphs and can be visualized in two dimensions [16].

With the increase in the size of commercial software in industry, the capabilities of 2D visualization become limited [16]. Thus, it is necessary for us to seek more efficient ways to represent software.

Improvements on graphic techniques and reduced price of graphic technology made 3D visualization more and more popular [13]. Compared to 2D visualization, 3D visualization makes more efficient use of screen space. It gives additional dimension to

encode more knowledge. It also provides much larger working volume and can handle much larger information load [16]. Furthermore, with good navigation mechanisms, 3D visualization provides user-intuitive exploration and interaction with the system [16].

In addition, the good use of intelligence amplification techniques allows a user to apply his or her "perceptual, cognitive and intuitive skills effectively within 3D environment" [16]. The designers of the Information Visualizer from Xerox PARC comment that "three-dimensional displays help shift the viewing process from being a cognitive task to being a perceptual task. This transfer helps to enable humans' pattern matching skills." Last of all, due to the similar nature of software objects and three dimensional objects, using 3D visualization allows the user to grasp the concept of object-oriented structure of large software more quickly and easily [16].

2.2.2. Visualizing Object-Oriented Software Structure

Object-oriented programming claims to give programmer more flexibility than the traditional procedural programming, its ease of modification makes it popular in large-scale software engineering [17]. Furthermore, proponents of object-oriented programming claim that

"Object-oriented programming is easier to learn for those new to computer programming than previous approaches, and Object-oriented programming approach is often simpler to develop and to maintain, lending itself to more direct analysis, coding, and understanding of complex situations and procedures than other programming methods" [17].

For these reasons, many of the visualization tools in industry are developed to represent the dynamics of object-oriented software in 3D space to provide more effective software analysis.

- **Object-Oriented Programming**

Object-oriented programming emphasizes the construction of objects and the relationship and communication between objects. Each of the individual objects is capable of receiving messages, processing data and sending messages to other objects.

Object-oriented systems can best be described as a collection of classes and *instances* (or *objects*) of those classes. Classes represent the definition of an object, its interface and information it may encapsulate. Instances are created from classes. Instances of the same class share an interface and functionality (*methods*), but not data (*properties*). Each instance of a class has its own properties, although these must all be of the same type.

- **Mapping Object Oriented System to 3D Visualization**

Most large-scale software is developed using the object-oriented paradigm. To visualize the object-oriented paradigm, we need to first decide how to map an object-oriented system into a 3D space to create useful visualization. There may not be a single best mapping which is appropriate in all circumstances. Certain situations or user requirements may dictate what the best mappings are.

A user may wish to view a system with the emphasis on class content rather than class hierarchy. In this situation, a suitable mapping may involve using many of the

available 3D properties such as dimension, color or transparency to represent software metrics such as number of methods or number of children as lines of code while the class hierarchy may be only very brief or not represented at all [16].

2.2.3. Using Visualization upon Software Metrics

Software metrics is the measurement of properties of a piece of software or its specification [20]. Software metrics is important because with software metrics, we are able to identify where the complexity of software can be reduced or where more testing should be applied. Software metrics allows the project manager to make estimates about the degree of project completeness, effectively assigns tasks and resources, and more importantly, identifies changes to be made so that software development and maintenance is less expensive and more time efficient [19].

So how can we tie software metrics into 3D visualization? Since both software metrics and software visualization reduce the complexity and detail of a given software system, by using software visualization to interpret software metrics instead of merely program source code, we hope to further reduce the complexity and speed up the process of software analysis [16].

Most 3D visualization tools commercially available to us are developed to merely represent the actual lines of code of the software. By contrast, our 3D visualization tool allows the user to display 3D objects as metrics of software modules and their relationships.

2.3. Existing 3D Visualization Tools

In this section we present some of the commercially available 3D software visualization tools. We will also discuss the similarities and differences between these tools and the 3D visualization tool we developed in this project.

2.3.1. Using the Metaball Metaphor to Visualize Source Code

According to Rilling and Mudar, *"Metaballs, also known as metablobs, soft objects, point clouds or more generally implicit surfaces, are a 3D object modeling technique which blends and transforms an assembly of particles with associated shapes into a more complex 3D shape, whose use is most suitable for animal and other organic forms"* [18]. Furthermore, *'Metaballs have found extensive use in representing and visualizing complex organic shapes and structural relationships such as DNA, humans, animals, and other molecular surfaces* [See Figure 2.2]. *Extensions include grouping of particles, selective influence over other particles, hiding particles, etc."* [18].

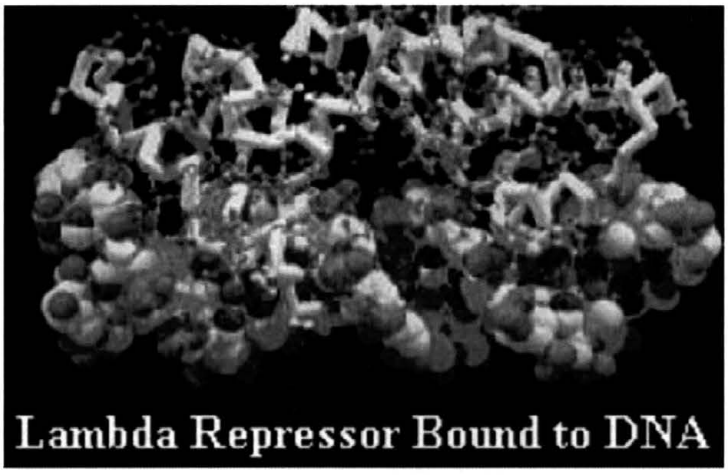

Figure 2.2

Rilling and Mudur defined a metaball *"by a three-dimensional variable density field, radiating from a given center point. The value of the field can vary linearly with distance from the center, or in any other way expressible via a mathematical formula"* [18].

Rilling and Mudur introduced the idea of using the metaball to represent objects within the software (See Figure 2.3). They propose that using the metaball metaphor to visualize software entities can be effective in software visualization: *"By defining visually intuitive mappings between the entities or parameters in the software slices, and metaball models, they can create a 3D virtual environment to emphasize the significantly influences the entity of interest, hide insignificant influences and zoom into entity-groups for understanding more detailed interactions"* [18].

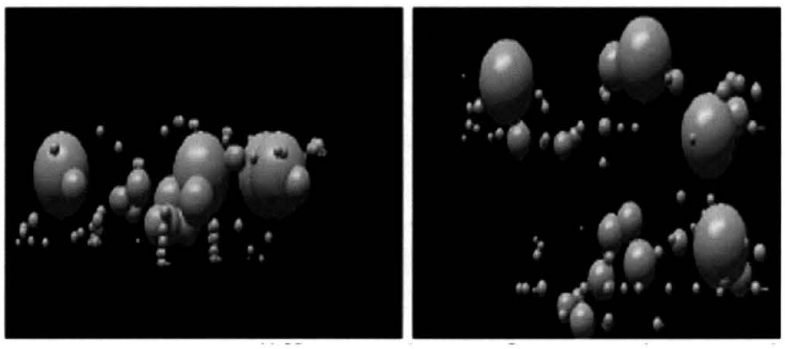

Figure 2.3

Below we make comparisons between the Metaball visualization tool and the 3D Software Metrics Visualization tool:

- The Metaball visualization tool uses properties of metaballs as a visualization metaphor to present program source code, while the 3D Software Metrics

Visualization tool uses the properties of spheres and three-dimensional planes as a visualization metaphor to represent metrics of software modules.

- The Metaball visualization tool uses the overlapping of the 3D objects to represent their relationships with each others, whereas the 3D Software Metrics Visualization tool uses line connections between modules to represent the modules' relationships with each other.
- In order to reduce visual complexity, both the Metaball visualization tool and 3D Software Metrics Visualization tool support the method of program slicing which is "a decomposition technique that transform a large program into a smaller one that contains only statements relevant to the computation of a selected function" [16], so that the user can visualize part of the system instead of the whole at one time.
- Both tools are embedded with a user navigation interface that allows users to navigate through the system in all directions.
- Unlike the 3D Software Metrics Visualization tool, the Metaball visualization tool does not provide the user the ability to generate a customizable view for a more effective software analysis.

2.3.2. The Sv3D Framework

"Sv3D is a software visualization framework that builds on the Seesoft metaphor" [14]. The Seesoft metaphor is a visualization metaphor which "allows one to analyze up to 50,000 lines of code and simultaneously by mapping each line of code into a thin row. The color of each row indicates a statistic of interest" [10].

Sv3D transforms the two-dimensional views from the Seesoft metaphor into three-dimensional representations by adding the third dimension, color texture, an abstraction mechanism that maps various artifacts of the software system and attributes into 3D objects (See Figure 2.4 and Figure 2.5) [14].

Sv3D separates data collection from visualization, which makes it independent of the analysis tool. "It accepts a simple and flexible input in XML format". The result is then displayed as a dynamic view containing 3D objects [14].

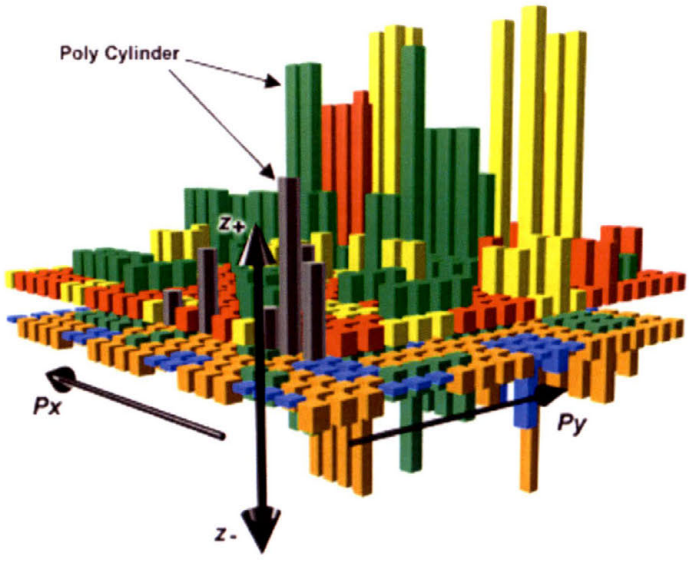

Figure 2.4

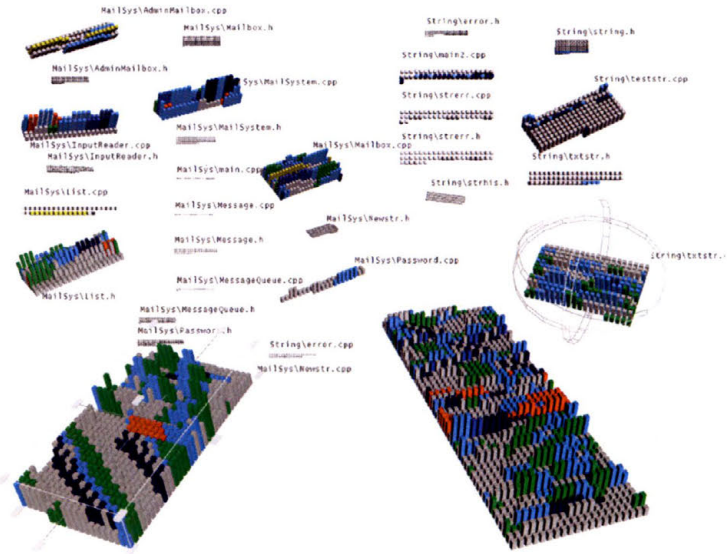

Figure 2.5

Below we make comparisons between the Sv3D tool and the 3D Software Metrics Visualization tool:

- The Sv3D tool only presents source code analysis, whereas 3D Software Metrics Visualization tool presents metrics analysis for any given hierarchical structured system.
- Unlike the 3D Software Metrics Visualization tool, Sv3D does not provide the ability to generate a customizable view for more effective software analysis.
- Both tools include a navigation interface that allows users to view the system in all directions.
- Both tools use the x, y, z coordinate system to measure properties of the software system.
- Both tools separate data collection from visualization and are independent of the analysis tool.
- Both tools use the method of applying texture, position, shape and size of the 3D object to represent attributes software modules or code.

2.3.3. Software Landscapes

In Software Landscapes: Visualizing the Structure of Large Software Systems, the authors use a "three-dimensional visualization technique that represents the static structure of object-oriented programs using landscape-like distributions of three-dimensional objects on a two-dimensional plane" [3].

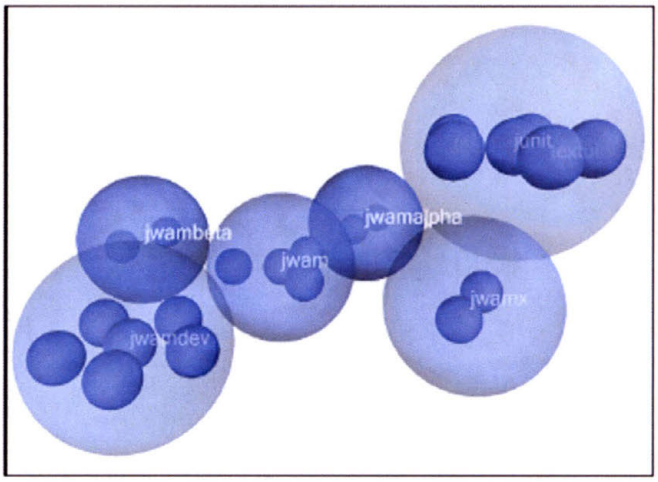

Figure 2.6

The system is based on the hierarchy of packages, classes, methods and attributes in the software system. The package is represented with nested spheres. The outermost sphere represents the root of the package, inside a second layer of spheres representing the package that are directly related to the root, and this continues with multiple layers of spheres nested within (See Figure 2.6). Transparencies are introduced so that a view into the system is possible [3].

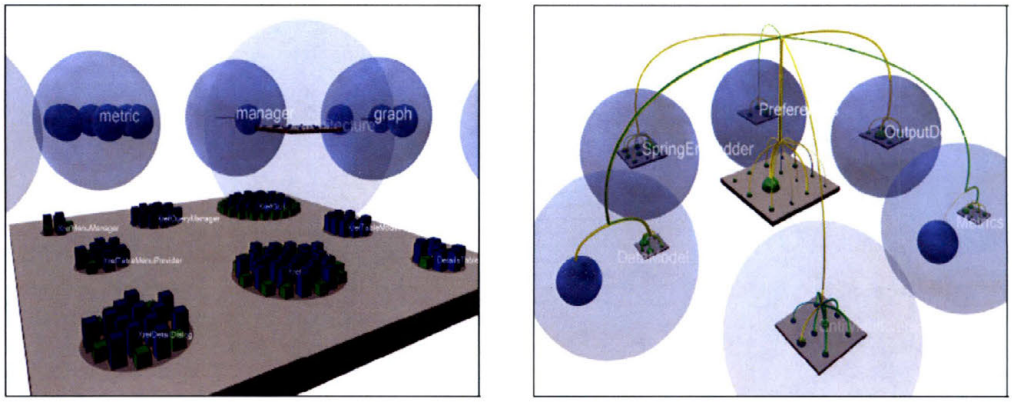

Figure 2.7

Circular discs are used to represent classes and methods, and within the discs there are solid cuboids that represent the attributes (See Figure 2.7). Relations between entities are represented by a direct line connecting both entities. The line has certain properties like color, brightness gradient and thickness which characterize the type, direction of the relation and quantity of presented connections. However, due to the large number of entities and relations, presenting all the necessary information is unmanageable in two-dimensional space. In order to solve the problem, the author turns the system into three-dimensional space and introduces a solution called the hierarchical net [3].

Below we make comparisons between the Software Landscape and the 3D Software Metrics Visualization Tool:

- Both tools employ hierarchical structure with the use of spheres and planes as a visualization metaphor.
- Both tools introduce transparency to 3D objects to reduce the overabundance of information.

- Both tools include a navigation interface that allows users to view through the system in all directions.
- The 3D Software Metrics Visualization tool shows the direct relations among software modules by using line connections. Software Landscape instead uses line connection to show intermediate relations between modules through a hierarchy net.
- Unlike 3D Software Metrics Visualization tool, Software Landscape does not provide the ability to generate a customizable view for more effective software analysis.
- Software Landscape does not separate data collection from visualization whereas 3D Software Metrics Visualization tool does.

CHAPTER 3

RESEARCH METHODOLOGY

3.1. 3D Software Metrics Visualization Engine Architecture

The overall structure of this software project can be categorized as three parts (See Figure 3.1). First of all, a back-end engine is used to retrieve essential data from given software and derives from them a set of software metrics. Examples like the size of the module, module complexity and its relationship with others are used. The generated metrics data are then stored into a text file format (See Section 3.2).

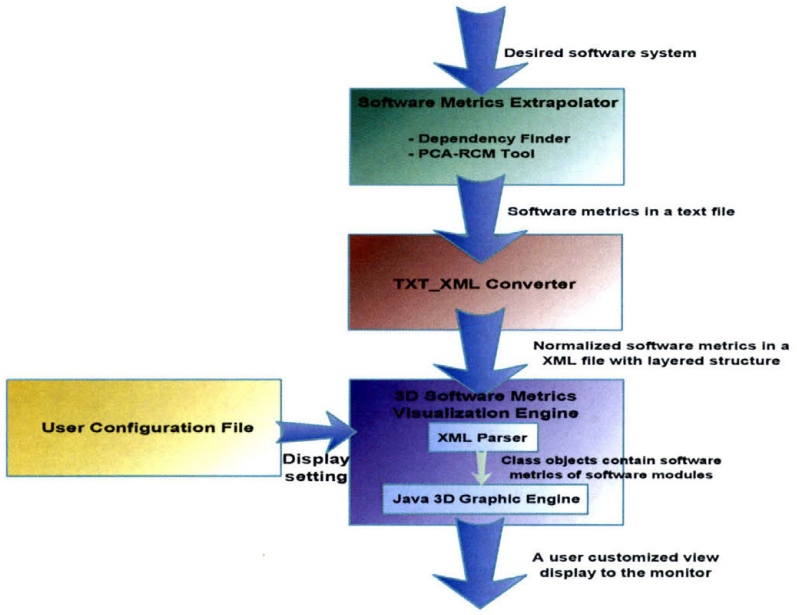

Figure 3.1

Next, we developed a simple C++ program, TXT_XML converter that reads the metric data from the text file, normalizes them, and exports them into a XML file with a layer structured format (See Section 3.3). The layer-structured format simulates the object-oriented paradigm where the metrics can now be viewed as attributes of modules in the software system. This will also allow our visualization engine to convert the metrics data into 3D objects in a faster and more manageable pace.

Now we import the XML file to our front-end visualization engine that is implemented in Java3D. An XML parser embedded within the visualization engine reads through the layer-structured metrics data and stores them as class objects. Finally, the class objects are displayed on the screen as 3D objects (See Section 3.4).

The 3D objects in the front-end visualization engine are categorized as artifacts and layers that correspond to the basic hierarchy of a software system or other layer-structured systems such as – file systems or network systems. Artifacts are represented by spheres with different volumes, color hues and color intensities that correspond to properties of class objects. Layers are represented as three dimensional rectangular planes. They are containers of artifacts. The layers (planes) also act as a coordinate system for the artifact. The location of each artifact can be used to indicate additional properties of class objects.

Before displaying the 3D object, a configuration file is available for users to manipulate. Users have the ability to customize different views of attributes from class objects when represented as 3D objects. The configuration file also provides users the option to turn on and off extra features embedded within the system. These features include the fan-in and fan-out relationships among software modules and a set of user

definable regions. The customizable view capability and extra features are implemented to accelerate the users' ability to analyze the software system.

In the next couple of sections, we will look at each part of this software project in detail.

3.2. Using Back-End Engine to Extrapolate Software Metrics

The back-end engine is merely a tool used for data collection. By separating the back-end engine from the main visualization engine, we now have the option to visualize any kind of system that is designed with a hierarchical structure. Hence, we won't limit ourselves to only being able to visualize one specific kind of software system.

The intention of this thesis is to illustrate that with the good use of the 3D visualization tool, effective software visualization is possible. However, finding and developing the best way of extracting software metrics is not our primary objective. On top of that, there are many remarkable commercial and academic tools that allow us to effectively extract useful software metrics from given software. Here we use Dependency Finder and PCA-RCM as our two primary software metrics extrapolators [9, 24]. Dependency Finder is a tool that generates software metrics, including lines of code, cohesion level, coupling level and software modules relationships that are considered as important metrics in software engineering (See Figure 3.2 and Figure 3.3). The PCA-RCM is a tool that collapses some of the software metrics mentioned above into a set of more meaningful and simpler software complexity metrics with the intention of providing more effective software quality analysis.

Here we provide a brief definition of a list of software metrics that are considered as important measurements in software engineering:

Source Lines of Code – "Source lines of code (SLOC) is a software metric used to measure the amount of code in a software program. SLOC is typically used to estimate the amount of effort that will be required to develop a program, as well as to estimate productivity or effort once the software is produced" [21].

Cohesion – "Cohesion is a measure of how well the lines of source code within a module work together to provide a specific piece of functionality." The level of "high cohesion" or "low cohesion" is normally being used to describe the robustness, reliability, reusability and understandability of a particular module in the software [6].

Coupling – "Coupling or dependency is the degree to which each program module relies on each other module. Coupling is usually contrasted with cohesion. Low coupling often correlates with high cohesion, and vice versa". "In object-oriented programming, coupling is a measure of how strongly one class is connected to another" [5].

Software Complexity - Software complexity can be further refined into the following components including functional complexity, fractional complexity and operational complexity. Finally, there is also a single relative complexity value that can be obtained for replacement during the software testing process [15].

3.2.1. Dependency Finder

Dependency Finder is developed by Jean Tessier to keep track of functional dependency of complex programs. The program extracts dependency graphs from a given java program and has the capability to compute object-oriented software metrics that provide an experiential quality assessment of the given code.

Dependency Finder derives object-oriented metrics like lines of code, unique operators and operands, functional cohesion and coupling values of the given software (See Figure 3.2 and Figure 3.3).

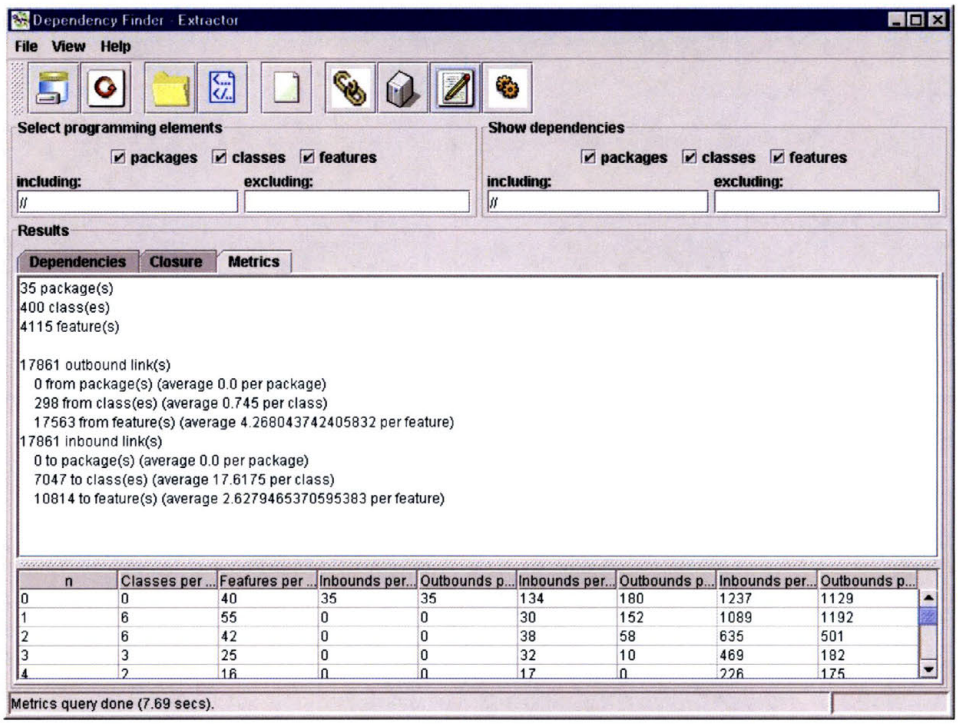

Figure 3.2

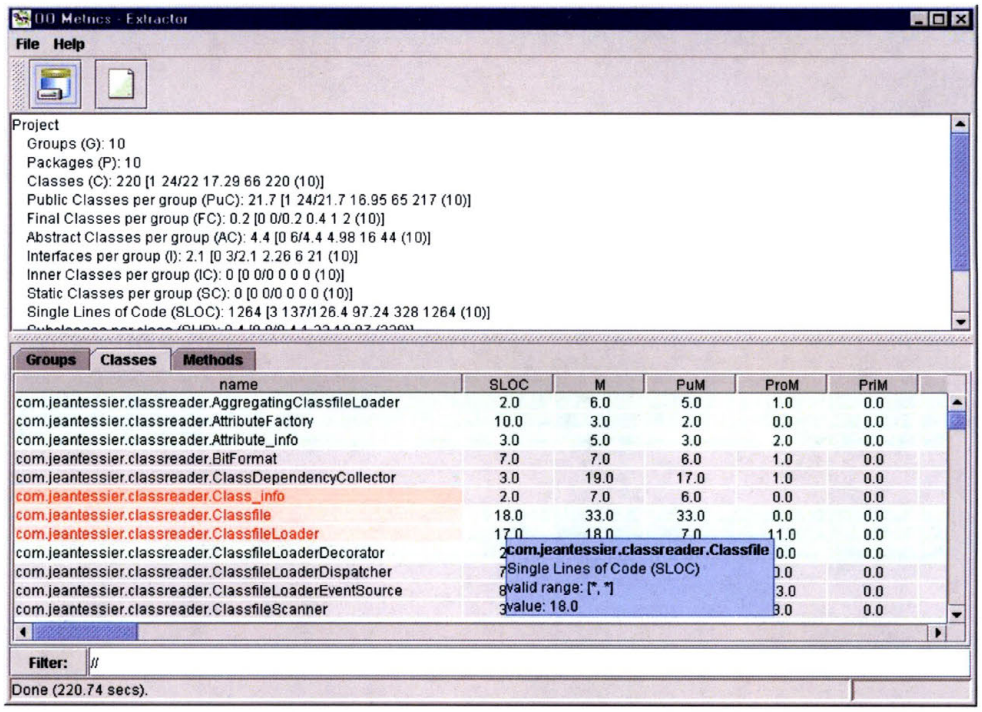

Figure 3.3

The benefits of using Dependency Finder are the following:

- Dependency Finder is an open source software and free [22].

- Dependency Finder comes in many forms, including command-line tools, a graphical user application and a web application that can be used from a browser [22].

- Dependency Finder can compute closures, that is, follow dependencies and find everything reachable from a given starting point. This can help package related components together, or verify code respects encapsulation [22].

3.2.2. PCA-RCM Tool

PCA-RCM is a tool developed by Gregory Hall at Texas State University-San Marcos, Department of Computer Science with the purpose of extracting complexity measurement from a given software system [15].

According to Khoshgoftaar and Munson, "Certain complexity metrics that have been shown to be distinctly associated with software faults." And according to research, "there are probably no more than four or five distinct complexity domains that are measured in some degree by each of the existing metrics." PCA-RCM provides us the ability to extrapolate functional complexity, fractional complexity and operational complexity domains from a pool of metrics of a given software system. In addition, it is possible to further collapse these complexity domains into a single relative complexity domain [15].

With these reduced and more meaningful complexity domains, along with other software metrics allowing users to choose to view within the visualization tool, the users now have more potential to make good analyses of software quality.

Module	DOMAIN1	DOMAIN2	DOMAIN3	RCM
test056 c/main	2 46694	1 78522	-1 69451	74 90414
test056 c/use_err	-0 97264	-0 20754	-0 34891	39 62502
test056 c/proc_file	0 21421	0 33424	-0 27779	52 34030
test056 c/get_token	0 26249	1 18992	2 20703	58 62880
test056 c/fil_chr	-1 02965	-0 07961	-0 38044	39 30154
test056 c/rdchr	0 84462	0 26401	2 12062	62 09891
test056 c/chk_token	1 75985	-3 01033	0 07897	60 57931
test056 c/put_token	0 13892	0 19222	-0 50463	50 94553
test056 c/chain_alpha	-0 06045	0 03713	0 64874	50 53607
test056 c/allo_id	-0 76540	0 15912	-0 40613	42 32114
test056 c/allo_rf	-0 66730	0 09584	-0 42600	43 09765
test056 c/add_rf	-0 98052	-0 04449	-0 41772	39 79093
test056 c/prnt_tbl	0 04875	-0 48107	-0 39747	48 79502
test056 c/prt_hdr	-0 65460	-0 04162	-0 02301	43 56804
Test056 c/nl	-0.60523	-0 19302	-0 17874	43 46759

Figure 3.4

Above is a sample text file that contains a complexity chart and can be displayed as attributes of 3D objects in our visualization engine (See Figure 3.4). The first column describes the name of the modules we want to analyze. The next four columns, domain1, domain2, domain3 and RCM, are associated with the functional complexity, fractional complexity, operational complexity and single relative complexity of the module, respectively.

3.3. TXT_XML Converter

The second stage of our visualization tool is the TXT_XML converter. The primary function of the TXT_XML converter is to streamline the software metrics data provided from the back-end engine into an object-oriented layer structure (See the example in Figure 3.5).

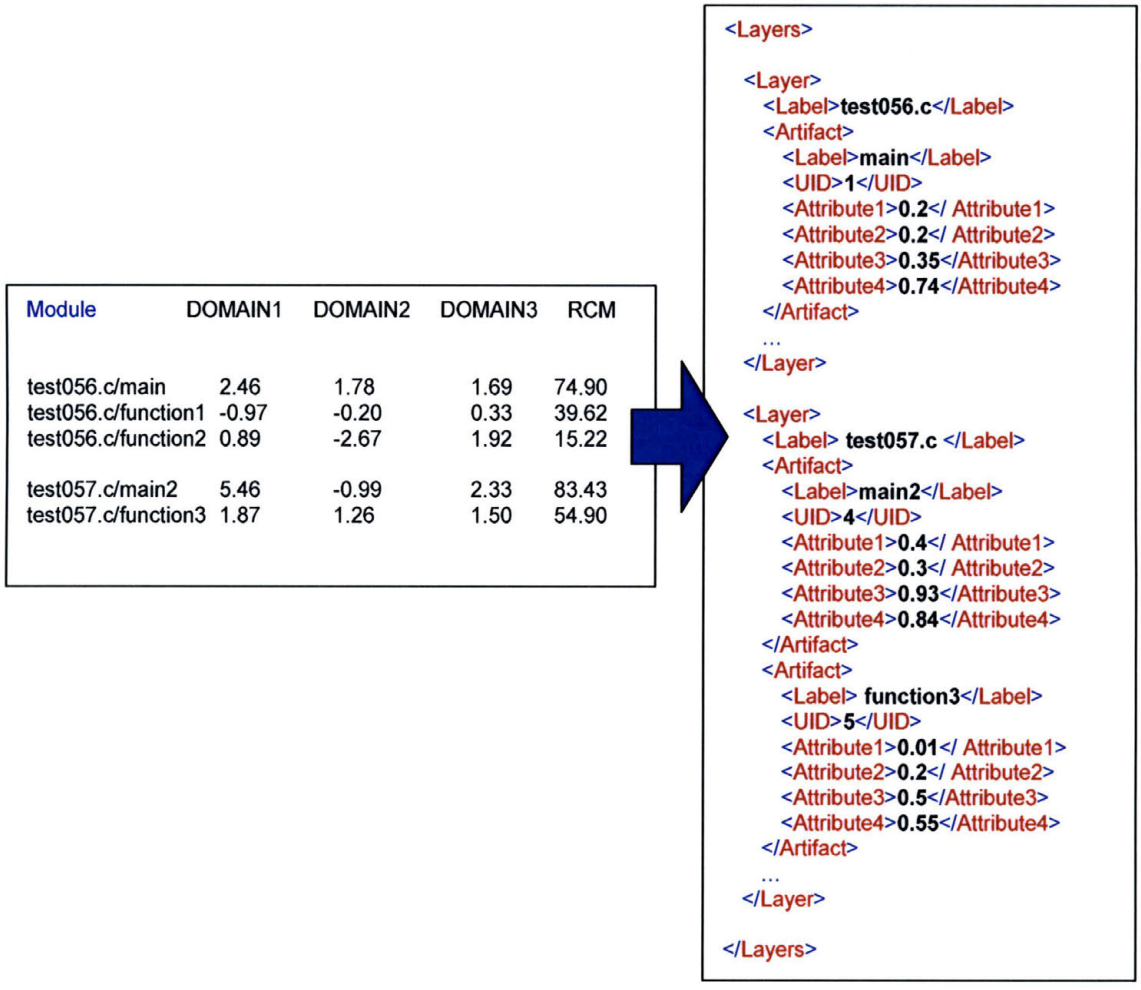

Figure 3.5

The above XML file is formatted into a layered structure. Artifacts are contained within the layer. An identification (UID) number is associated with each layer to maintain consistency. Artifacts are also assigned an identification (UID) number and a set of attributes. The artifact attributes are associated with the software or system metrics values provided by the back-end engine. The numeric values of these attributes have been normalized between 0 and 1 to allow a more convenient and maintainable transaction between the TXT_XML converter and the front-end graphical engine when we transform these metrics data into 3D objects.

3.4. Front-End 3D Graphical Engine and Interface

The 3D Software Metrics Visualization Engine we designed in this project is a front-end graphical engine developed under Java3D, a 3D graphic development tool. The engine includes an XML parser which imports normalized and layer structured metrics data from the XML file, which is generated by the TXT_XML Converter. An XML parser is then used to turns these formatted metrics data into class objects. By using Java3D, we map the metrics data into 3D objects. These 3D objects consist of simple spheres and planes (See Figure 3.6).

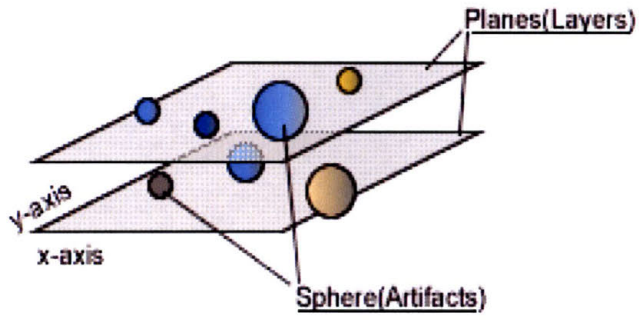

Figure 3.6

Planes – Planes that we call layers contain artifacts. Depending on the situation, planes can be used to represent different functionalities of a task; or an overall structure of classes when associated with a software program, or they can be used to represent different file directories when we visualize a file system. Each plane associated with a layer identification number, Layer ID, and the name of the layer will be shown as a label on the top left-hand corner of the layer.

Spheres – The spheres that we name artifacts generally represent as software metrics properties we derived from the given software or system. Again depending on the situation, spheres can be used to represent a specific method within the class from a software or can be used to represent a specific file within a directory of a file system. The sphere contains an artifact identification number – artifact ID. It is defined by a label, volume, color hue, color intensity and location within the planes. These properties are directly associated with the attribute of the class object in the software or system. Figure 3.7 shows us an example of the final display after the artifacts and planes are generated and presented to the user.

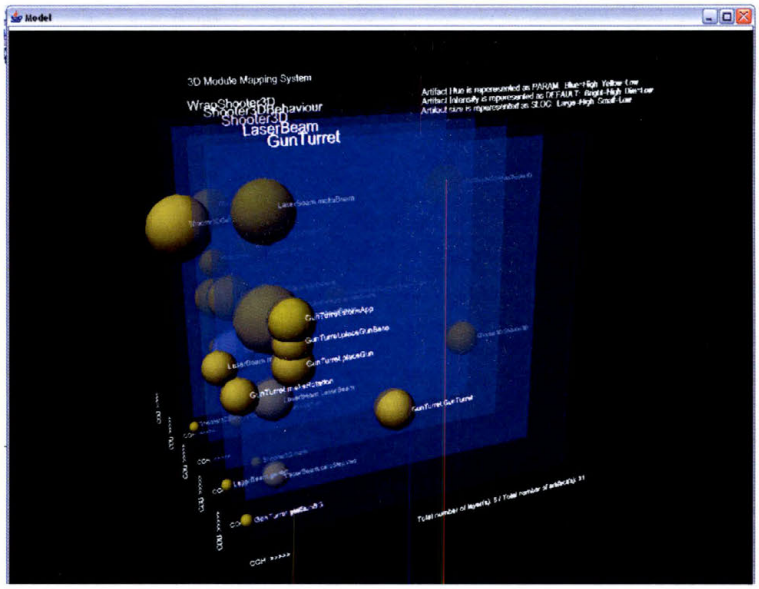

Figure 3.7

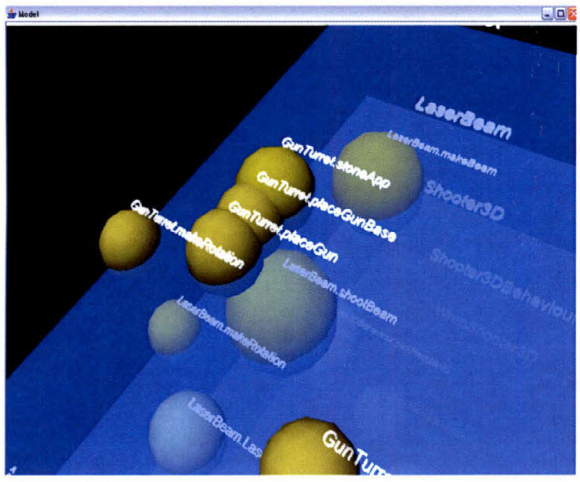

Figure 3.8

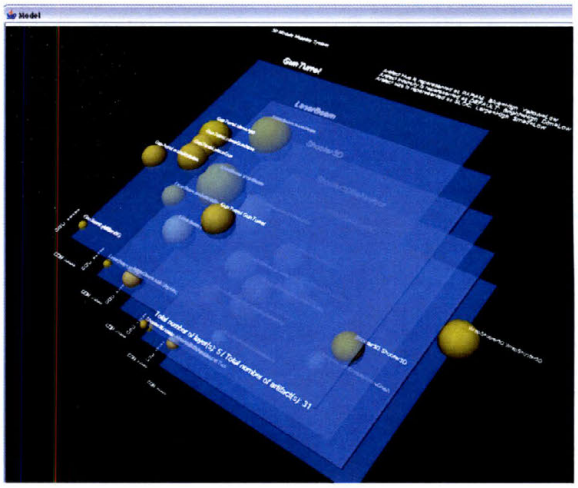

Figure 3.9

The interface allows users to navigate through the entire system with a 360 degree view (See Figure 3.8 and 3.9). The flexible navigation provides the user a dynamic view, with the potential of increasing the accuracy and minimizing the time span the user needs to retrieve crucial information from the interface to understand the software. Figure 3.10 shows the basic navigation for our 3D Metrics Visualization Engine.

Key	Movement	Alt + Key
Left arrow	Rotate left	lateral translate left
right arrow	Rotate right	lateral translate right
up arrow	move forward	
Down arrow	move backward	
PgUp	rotate up	translate up
PgDn	rotate down	translate down
+	restore back clip	
-	reduce back clip	
=	Restore view	
Left Mouse Button Drag	Rotate any direction	
Right Mouse Button Drag	translate any direction	

Figure 3.10

3.5. User Configuration File for Customizable Views

Before we display the 3D visualization objects onto the screen, the user has the option of changing a set of settings from a configuration file The file contains instances that are represented as the metrics data of the software By changing the setting in the configuration file, the user is now given the ability to customize a desired view regarding an aspect of the software.

The instances in the configuration file are represented as integers that correspond to the metrics that represented as 3D object properties in the final output. For example, the x-axis has an integer '1', the y-axis has a value of '2', the artifact has a value of '3' and the color hue has a value of '4'. In the 3D engine, the x-axis of the plane now represents the attribute one of the artifacts, the y-axis of the plane represents attribute two of the artifacts and so on. (See Figure 3 11).

```
xaxis          1
yaxis          2
aritfactsize   3
colorhue       4
intensity      5
relationship   0
helperregion   0
```

Figure 3.11

Here we map a simple program into the 3D Software Metrics Visualization Engine using settings of the user configuration file shown at Figure 3.11. The user now produces a view that is shown in the image below (See Figure 3.12). The x- and y-axis of the system represent attribute 1 and attribute 2 of the modules within the program. The size of the spheres represents attribute 3 of the modules. The color hues and intensities of the spheres represent attribute 4 and attribute 5 of the modules.

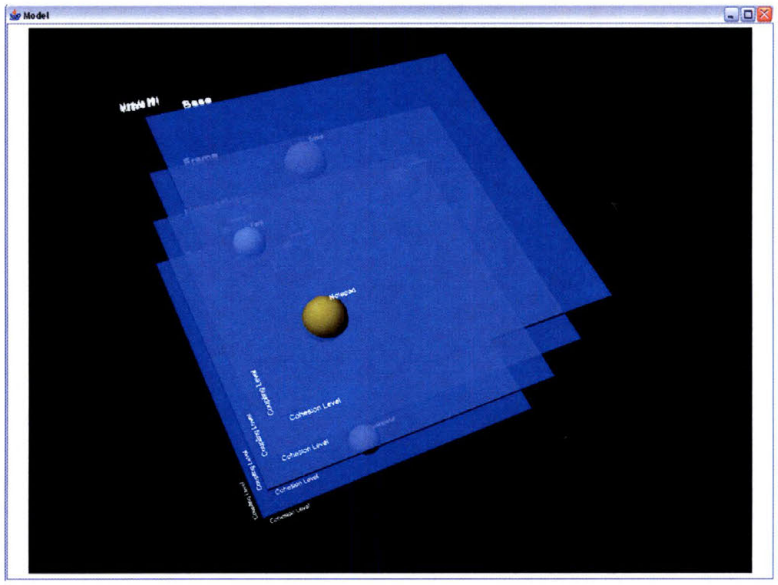

Figure 3.12

Next the user maps the same program into the 3D Metrics Visualization Engine using the user configuration file setting show in Figure 3.13. This time every aspect of the artifact (sphere) is measured as attribute 1 of the software module.

xaxis	**1**
yaxis	**1**
aritfactsize	**1**
colorhue	**1**
intensity	**1**
relationship	**0**
showregions	**0**

Figure 3.13

Although producing such a view (See Figure 3.14) might not have significant meaning for users when trying to understand the architecture of the program, it shows the flexibility the engine provides, which is that it is able to produce a view depending on what users may think is the best view for them.

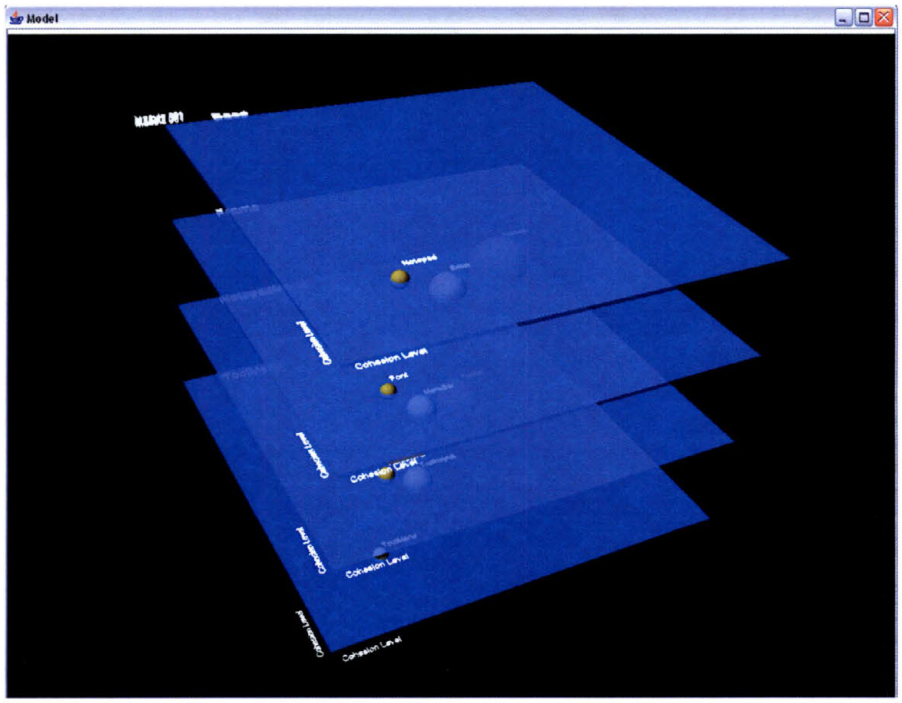

Figure 3.14

3.6 Additional Features for Enhancing Analysis

The 3D Metrics Visualization Engine also includes other mechanisms that aid the user when analyzing software systems. First of all, it gives the user the option to choose whether or not to show the connections between modules by changing the setting of the **relationship** in the user configuration file. Being able to see these connections is sometimes significant. Let us say the user decided a certain module is developed poorly or rather complex and needs to be refined. Now, if this module is connected to many other modules throughout the system, then changing it might introduce further instability and defects to the systems. However, other times it is better for the user to turn off these connections among the modules in order to reduce the visualization burden. Figure 3.15 shows some of the line connections among modules. The blue color end of the line represents the connected node as a source or parent node. The green color end of the line represents the connected node as a destination or child node.

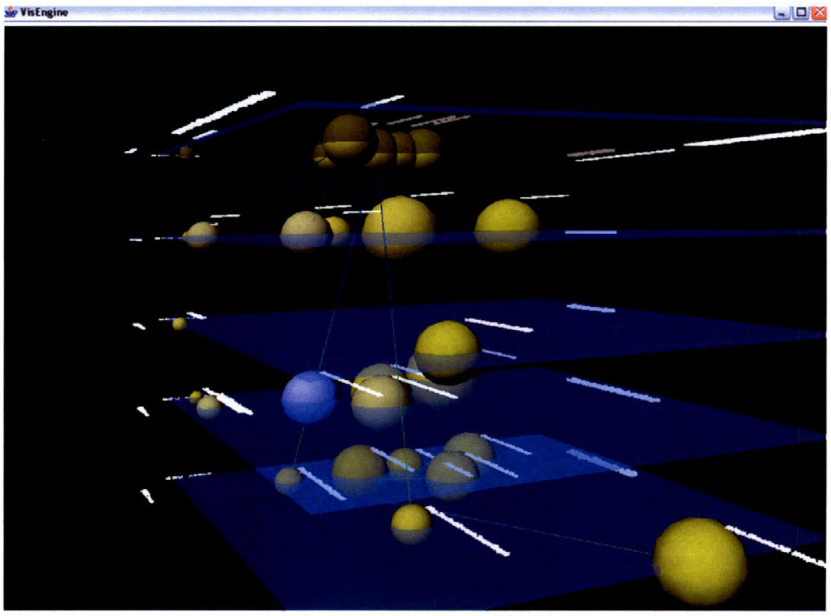

Figure 3.15

Next, the 3D Metrics Visualization Engine provides the user a set of definable colored regions. The colored regions can be turned on or off by changing the setting of the **showregions** in the user configuration file. These colored regions are intended to aid the user when trying to quickly determine which modules are considered as problematic and which modules are considered as well-developed. These regions contain lower- and upper-bound threshold values in both x and y directions of the planes and are drawn onto the layers based on those threshold values.

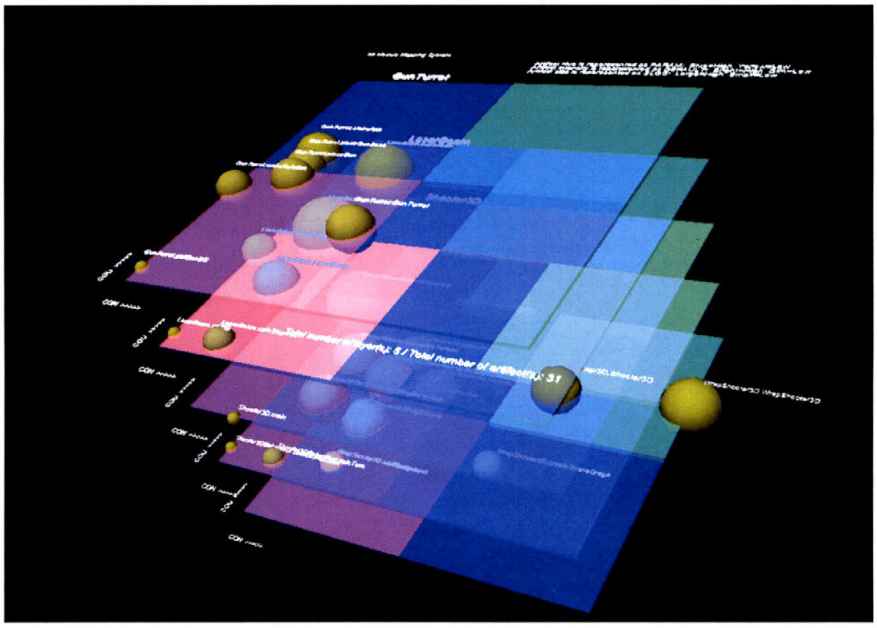

Figure 3.16

In the example above (See Figure 3.16), red and green regions are placed within each layer. In this case, artifacts that fall within the red colored regions can be categorized as software modules with low cohesion and high coupling levels. These modules in software engineering tend to be problematic and hard to maintain; whereas, the artifacts that are represented as well-developed modules with high cohesion and low coupling levels fall into the green colored regions.

3.7. Problem of Displaying a Large Software System at Once

Regardless of what software visualization tool or technique a person uses, when the given software system is enormous, analyzing the entire system at once is daunting and sometime impossible.

The key to resolve this issue is to analyze the software system one part at a time. Knight and Munro mention that introducing multiples views of data that suit the user's need at the time can be a successful approach to analyzing software [13].

In our visualization tool, before a user imports the raw software metrics data from the back-end software metrics extrapolator to TXT_XML Converter, he or she has the option to choose which set of views he or she wants to analyze at the final output. This gives the user the flexibility to view the entire system as a whole or to view parts of the system at a time.

CHAPTER 4

RESULTS AND ANALYSIS

4.1. Experiment Objectives

An experiment is conducted in order to identify the usefulness of our visualization tool, to analyze the quality of given software. To set up the experiment, we first need to determine our target audiences. In order to have sufficient knowledge to analyze any basic software systems, we conclude that our test subjects should have a basic understanding of the data structure used in common software programming.

After we determine our target audiences, two survey tests are being setup: the 3D visualization test and the standard test. The 3D visualization test is used to analyze a software system with single or multiple levels of hierarchies using our visualization tool. A view with 3D objects containing information in the software metrics about the system are presented to the test subject.

The Standard test is used to analyze a software system with single or multiple levels of hierarchies using no visualization aid. A text file containing information about the software metrics in the system is presented to the test subject.

By using the null hypothesis, we try to show that with or without 3D visualization support, there is a difference in both accuracy and time efficiency of how well a person perform on software analysis.

"The null hypothesis, H_0, represents a theory that has been put forward, either because it is believed to be true or because it is to be used as a basis for argument, but has not been proved" [8].

"The alternative hypothesis, H_1, is a statement of what a statistical hypothesis test is set up to establish [8].

The Statistics Glossary v1.1 says that "we give special consideration to the null hypothesis. This is due to the fact that the null hypothesis relates to the statement being tested, whereas the alternative hypothesis relates to the statement to be accepted if / when the null is rejected." The final conclusion of the test is that we either "Reject H_0 in favor of H_1" or "Do not reject H_0". Whether or not we reject the null hypothesis, we, however, can never conclude "Reject H_1" or even "Accept H_1" [8].

Based on the above definitions we state the following in our case:

H_0: There is no difference between subjects who participate in the 3D visualization test and the standard test regarding the accuracy and efficiency of the survey result.

H_1: Subjects who participated in the 3D visualization test had better results in both accuracy and efficiency while trying to analyze a software system than they did when taking the standard test which provides no visualization aids.

4.2. Experiment Requirements

Below is a list of requirements that were needed before the surveys were being carried out.

Hardware Requirements:

- A group of high performance desktop computers with high-end graphic cards that have Windows or Linux operating systems installed
- A timing device

Software Requirements:

- Java Platform Standard Edition
- Java3D standard libraries
- 3D Software Metrics Visualization Engine

Finally, we needed a pool of participants to take part in the surveys.

4.3. Experiment Procedures

The survey was set up using the following procedure. A group of university students who majored in Computer Science and who were also enrolled in courses such as Networking, Operating Systems and Programming Language were selected as our target audience. Based on the fact that most of them already had a course in Data Structures, we supposed that they understood the basic principles and structure of software programs.

Next, we separated our participants into two groups; each group took part in two different tests that were composed of a variety of software engineering questions. Participants also were provided with a set of instructions about how to take the test and how to use our visualization tool (See Appendix A.1. and Appendix A.2.).

Each survey was specifically related to one of the two programs; we called them program A and program B. The difference between program A and program B is that program A is a relatively small program containing only a single file with multiple function calls, and program B is a larger program containing multiple files with multiple function calls. By using a software metrics extrapolator like Dependency Finder and PCA-RCM, we acquired a set of metrics data for both programs. The metrics data were then either fed into the 3D visualization engine as 3D objects or into a text file as a list of numbers.

Next, group one took a test regarding program A with 3D visualization support and group two took a test regarding program B with no visualization aids. Subjects in both groups recorded the time used to take the test. Then group one took a test regarding program B that with no visualization aid, and group two took a test with

program A with 3D visualization support. Again, subjects in both groups recorded the time used upon the tests (See Figure 4.1).

Group 1 with visualization test on program A	**Group 1 with standard test on program B**
Group 2 with standard test on program A	**Group 2 with visualization test on program B**

Figure 4.1

4.4. Experiment Results

Accuracy and efficiency were the two primary measurements of our survey results. Tables below show the results we gathered after we examined each of the participant's performance upon the test. Figure 4.2 shows the participants' accuracy upon analyzing the given program with or without the aid of the visualization tools. The number represents the total correct answers out of the ten questions given in the survey test.

Program	Visualization	
	Yes	No
A	7	9
	10	6
	7	7
	9	4
	7	4
	7	4
	3	5
	6	5
	9	9
	7	7
	9	4
	3	6
	10	9
	8	10
	6	9
	7	4
	7	10
	7	8
	6	10
	6	7
	6	4
	6	9
	6	6

Program	Visualization	
	Yes	No
B	7	10
	10	4
	9	3
	9	5
	6	5
	7	3
	7	4
	5	5
	4	7
	7	5
	4	7
	7	4
	5	5
	10	5
	7	4
	9	5
	6	4
	7	5
	10	3
	9	5
	6	3
	10	3
	7	3

Figure 4.2

Figure 4.3 and shows that the participants' efficiency depended on whether or not they were provided with the aid of the visualization tool. The number represents the time, measured in seconds that it took surveyors to finish the ten questions given in the survey test.

Program	Visualization	
	Yes	No
A	540	840
	2100	1320
	780	660
	600	540
	1800	1200
	660	300
	180	240
	600	600
	1440	733
	4500	1032
	506	600
	240	347
	1560	840
	2700	423
	1200	780
	1500	900
	3360	826
	4800	456
	1620	816
	1500	300
	4800	406
	1980	913
	3180	430

Program	Visualization	
	Yes	No
B	960	1680
	900	300
	960	780
	660	600
	1500	600
	1140	600
	960	360
	600	600
	620	1080
	384	1800
	900	438
	371	660
	960	480
	570	600
	1500	600
	2220	1500
	756	600
	303	1800
	609	720
	1380	600
	294	1380
	1106	960
	540	1200

Figure 4.3

4.5. Analyzing Results Using Multifactor Analysis of Variance

In order to make a meaningful analysis of our results, we use existing statistical methods.

Based on the way that the experiments were being conducted, we introduced two independent variables and two dependent variables from our experiment. They can be categorized as the following:

Independent Variable:

- Program A or Program B
- With visualization or without visualization

Dependent Variable:

- Time used to finish the survey test
- How many correct answers participant scored on the test

Next, we used a statistical method named multifactor analysis of variance (ANOVA) to analyze our result. But first let us describe some terms used in statistics, so that the reader can understand how to read these terms within our analysis report.

F-distribution is "defined in terms of two independent chi-squared variables. Let u and v be independently distributed chi-squared variables with u_1 and v_1 degrees of freedom, respectively. Then the statistic: $F = (u/u_1) / (v/v_1)$ has an F distribution with (u_1, v_1) degrees of freedom" [11].

Since the F-distribution involves a ratio of sample variances of σ_1^2 and σ_2^2, and if the ratios is differ too much from 1 then we can reject $\sigma_1^2 = \sigma_2^2$ and thus reject the null hypothesis [7].

The P-value, or probability value," of a statistical hypothesis test is the probability of getting a value of the test statistic as extreme as or more extreme than that observed by chance alone, if the null hypothesis H_0, is true…"

> *"Small p-values suggest that the null hypothesis is unlikely to be true. The smaller it is, the more convincing is the rejection of the null hypothesis. It indicates the strength of evidence for say, rejecting the null hypothesis H_0, rather than simply concluding "Reject H_0" or "Do not reject H_0"* [9].

ANOVA stands for analysis-of-variance: "A statistical method for making simultaneous comparisons between two or more means; a statistical method that yields values that can be tested to determine whether a significant relation exists between variables" [2].

There are many statistical products that can be used to calculate ANOVA. Here we used a program called "Analyse-it" [1]. "Analyse-it" is able to import data from a simple excel sheet and then offer us a 2-way ANOVA test that provides us useful information such as the difference in mean between the samples, F-distribution and p-value.

analysed with: Analyse-it • General 1.73

Test	2-way between subjects ANOVA		
Comparison	Data by Program, Visualization		
Performed by	defstudent	**Date**	4 April 2006

n	92

Program	n	Mean	SD	SE
A	46	6.848	2.011	0.2965
B	46	5.978	2.206	0.3252

Visualization	n	Mean	SD	SE
Yes	46	7.109	1.829	0.2696
No	46	5.717	2.228	0.3285

Source of variation	SSq	DF	MSq	F	p
Program	17.391	1	17.391	4.34	0.0400
Visualization	44.522	1	44.522	11.12	0.0012
Within cells	356.391	89	4.004		
Total	418.304	91			

Figure 4.4

The chart above (See Figure 4.6) shows the result of comparing the accuracy between the two tests by either using or not using the visualization tool from a 2-way ANOVA. Here are the facts presented to us:

- The accuracy measurements of the F-distribution between the programs and with or without visualizations are both very different from 1.
- The accuracy measurements of the *p*-value between the programs with or without visualizations are both extremely small.
- The average mean of correctness of a person who used our visualization tool on a program was higher than that of a person who had no visualization aid.
- The average mean of correctness of a person who took a test with program A was higher than a person who took a test with program B.

Based upon the fact that the F-distribution (11.12) highly differed from the value 1 and the extremely small value of *p*-value (0.0012), we can now reject with high confidence that the null hypothesis stating that there is no difference between using or not using a visualization tool when measuring the accuracy of a person analyzing software. Furthermore, from looking at the difference between the means, we can also say that a person who used our visualization tool in software analysis scored more accurately than a person who did not use our visualization tool.

analysed with: Analyse-it • General 1.73

Test	2-way between subjects ANOVA		
Comparison	Data by Program, Visualization		
Performed by	defstudent	**Date**	4 April 2006

n	92

Program	n	Mean	SD	SE
A	46	1253.217	1175.282	173.2859
B	46	872.413	457.591	67.4681

Visualization	n	Mean	SD	SE
Yes	46	1355.196	1154.204	170.1781
No	46	770.435	397.287	58.5768

Source of variation	SSq	DF	MSq	F	p
Program	3335274.880	1	3335274.880	4.66	0.0336
Visualization	7864741.315	1	7864741.315	10.99	0.0013
Within cells	63715773.663	89	715907.569		
Total	74915789.859	91			

Figure 4.5

The chart above (See Figure 4.5) shows us the result of comparing the efficiency between the two tests by using and not using a visualization tool from a 2-way ANOVA. Here are the facts presented to us:

- The efficiency measurements of the F-distribution between the programs with or without visualizations are both very different from 1.
- The measurements of the *p*-value between the programs with or without visualizations are both extremely small.
- The average mean of correctness of a person who used our visualization tool on a program was higher than a person who had no visualization aid.
- The average mean of correctness of a person who took a test with program A was higher than a person who took a test with program B.

Again with the F-distribution (10.99) highly differing from the value 1 and the extremely small value of *p*-value (0.0013), we can now reject with high confidence the null hypothesis stating that there is no difference between using or not using a visualization tool in measuring the efficiency of a person analyzing software. However, by looking at the difference between the average means, we noticed this time that a person who used our visualization tool in software analysis performed a lot slower than a person who did not use our visualization tool.

The overall result shows us that a person who uses our visualization tool performs more accurately but less efficiently than a person who does not use our visualization tool. So what does that mean? We determine that our experiment may not be sufficient to tell us whether or not our visualization tool is truly effective after all. Though, a person was more accurate results when he or she uses our visualization tool, there is a with the significant time increases in using our visualization tool. However, we introduce the idea that if a person takes longer to analyze software, wouldn't he or she perform better?

4.6. Future Surveys

When we set up our survey, there were hindering factors that we did not take into consideration that could have led to the failure of the experiment. First of all, we never took into account the time required for the user to adapt to our visualization tool. When being introduced to new software, most likely a person will take a while before he or she is comfortable using the interface. Thus, recording the first time performance of the person using our visualization tool was a bad idea.

Secondly, we did not take human factors like the fact that most people do not read instructions into account. Even though the survey clearly states that a person should not use more than fifteen minutes on each test, the test results have shown us that a person took about an average of twenty minutes in program A and twenty-two minutes in program B, the times are both well over fifteen minutes. We believe that since the survey was assigned as a class project, most students worried that their performance on the survey would affect their actual class performance. Hence, students would rather get the answer right instead of taking the survey correctly.

In order to truly test the effectiveness of our visualization tool, we believe that further experiments need to be carried out with more careful planning.

CHAPTER 5

CONCLUSION

5.1. Goals Revisited

The goals of this research have been the following:

- To address problems regarding software visualization
- To discuss existing visualization schemas and tools for software analysis
- To design and implement a dynamic 3D visualization tool by adapting of existing approaches to create more effective software visualization
- To examine the effectiveness of our 3D visualization tool using surveys and statistical analysis

In the first and second chapter we address the problem that software products have grown larger and more complex then before. It is believed that, with traditional two-dimensional software visualization, effective software analysis is simply not possible. However, with the advancement in technology, the introduction of three-dimensional visualization, and the idea of using an intelligence amplifier and visualization metaphor, constructive software visualization can be achieved.

In chapter two we examine some of the software visualization techniques available to us. We discuss the benefits of using software metrics and object-oriented

programming. We review several commercially available tools such as the metaball visualization tool, Sv3D and Software Landscape and compare these tools with our visualization tool.

In chapter three we describe in detail each design and implementation stage of our 3D visualization tool. We emphasize that, by separating the procedure of data gathering and graphic visualization, we are not limiting ourselves to only making an effective visualization for one specific kind of software system; instead, we now are able to visualize upon any system that has a hierarchical structure. We also introduce the advantages of allowing the user to create customizable views to maximize his or her ability to analyze software.

In chapter four we conduct a survey of university students, and by using a statistical method, we try to prove that our visualization tool is indeed useful. However, due to hidden factors which we did not foresee, our results can not successfully determine the true effectiveness of the visualization tool. However, by examining our results, we can determine some of these hidden factors and how they hampered our experiment.

5.2. Future Work

Throughout the course of this thesis, the emphasis has been on adapting the advantages of the existing visualization techniques. We developed a customizable 3D visualization tool for effective software analysis. This has meant a lot of scope for experiments. As such, the experiments conducted by ourselves have only barely scratched the surface. There exist a potentially large number of further experiments which could be conducted by the user either with the existing architecture of the visualization tool or by modifying the architecture of the tool as suggested in section 5.2.1.

Future work may also entail providing a more user-friendly interface and adding extra useful features to the architecture. Some of the possibilities are suggested in sections 5.2.2.

5.2.1. Further Experiments

As we discussed in chapter four, the experiment we carried out has many hidden factors that we did not foresee, problems like the large amount of time taken for the participants to adapt to our visualization tool, the human factor that the participants did not bother to read the instructions before taking the survey. More future experiments need to be carried out in order to make meaningful determination of whether or not our visualization tool is truly effective. Here are some suggestions that may help us reduce bias in future experiments.

First, it will be helpful to embed the survey test into a time-tracking program which automatically keeps track of the time spent on each of the questions and the time spent on the individual survey as a whole. The program should automatically stop the participant when a limited time is allowed for the test. This is to prevent a participant from spending too much time in the survey, so he or she can get every question right.

Second, the survey may be required to be carried out multiple times with the same set of participants. This is because it takes a new user some time before he or she adapts to using the interface. Therefore, introducing the survey multiple times to the same subject and only taking the last set of results could maybe provide more meaningful suggestions about the effectiveness of our visualization tool.

5.2.2. Enhancing Interface Usability

Although the 3D visualization engine we developed is fully functional, it can be improved upon in the following ways:

Instead of having a separate user configuration file, the user configuration option should be embedded into the visualization engine so that a user does not have to exit the software and re-execute the program each time for a different view. In addition, we could provide the option of splitting the computer screen into multiple quadrants so that multiple views can be represented onto the screen simultaneously.

More user interactions can be added into the engine -- for example, features such as interactive mouse events. By using the mouse-over option upon individual modules, a user is able to see only its relation with other modules instead of showing every other relation among all the modules at once. This can further decrease complexity and reduce visualization encumbrances.

Animation can also be introduced. For example, if the user wants to modify a particular module within the system, then how would this affect the entire system? With animation, we can simulate the life cycle of a particular software after certain events take place, we can see how the software modules adapt to changes and evolve. This information can provide a tremendous amount of capability for a user to further understand how the given software would react to changes.

REFERENCES

[1] Analyse-It [Statistical software], Analyse-It Software, Ltd., Leeds, England, Retrieved from: http://www.analyse-it.com/company/contact.asp

[2] "ANOVA," *WordReference.com English Dictionary,* Princeton University, Retrieved from: http://www.wordreference.com/definition/ANOVA.htm

[3] Balzer, M., A. Noack, O. Deussen, and C. Lewerentz (2004) "Software Landscapes: Visualizing the Structure of Large Software Systems," *Joint EUROGRAPHICS – IEEE TCVG Symposium on Visualization,* Department of Computer and Information Science, University of Konstanz, Germany and Software Systems Engineering Research Group, Technical University, Cottbus, Germany.

[4] Brooks, F. P. (1987) "No Silver Bullet," *IEEE Computer,* pp. 10-19.

[5] "Coupling (Computer Science)," *Wikipedia, the free encyclopedia,* Retrieved from http://en.wikipedia.org/wiki/Coupling_%28computer_science%29

[6] "Cohesion," *Wikipedia, the free encyclopedia,* Retrieved from: http://en.wikipedia.org/wiki/Cohesion

[7] Devore, L. J. (2000) "F-distribution," *Probability and Statistics for Engineering and the Science.* California Polytechnic State University. San Luis Obispo, CA, pp 108-109.

[8] Easton, V. J., and J. H. McColl (1997) "Hypothesis testing," *Statistics Glossary* version1.1, Retrieved from: http://www.stats.gla.ac.uk/steps/glossary/hypothesis_testing.html

[9] Easton, V. J., and J. H. McColl (1997) "Hypothesis testing: P-Value," *Statistics Glossary* version1.1, Retrieved from:http://www.stats.gla.ac.uk/steps/glossary/hypothesis_testing.html#pvalue

[10] Eick, S. C., J. L. Steffen, and E. E. Summer (1992) "Seesoft - A Tool for Visualizing Line Oriented Software Statistic," *IEEE Transactions On Software Engineering*, vol. 18, no. 11, IEEE Press, Piscataway, NJ.

[11] Johnston, J., and J. DiNardo (1997) "F-distribution," *Econometric Methods,* Fourth Edition, McGraw Hill Co. New York, NY, pp. 530-531.

[12] Kiskinen, J. (2003) "Software Maintenance Costs." *Information Technology Research Institute,* University of Jyväskylä, Finland, Retrieved from: http://www.cs.jyu.fi/~koskinen/smcosts.htm

[13] Knight, C., and M. Munro (1999) "Visualizing Software – A Key Research Area," Visualization Research Group, Research Institute in Software Evolution, Department of Computer Science, University of Durham, Durham, UK.

[14] Marcus, A., L. Feng, and J. Maletic (2003) "3D Representations for Software Visualization," ACM Press, New York, NY, Department of Computer Science, Kent State University, Kent, Ohio.

[15] Munson, J. C., and G. A. Hall (1996) "Estimating Test Effectiveness with Dynamic Complexity Measurement." *Empirical Software Engineering*, Computer Science Department, University of Idaho, pp. 279-305.

[16] Nganou, A., R. A. Amad, S. Shi, and Xiaohoua X. "Software Visualization 2D versus 3D," The Department of Computer Science, Concordia University.

[17] "Object-Oriented Programming," *Wikipedia, the free encyclopedia*, Retrieved from: http://en.wikipedia.org/wiki/Object-oriented_programming

[18] Rilling, J., and S. P. Mudur (2002) "On the Use of Metaballs to Visually Map Source Code Structures and Analysis Results onto 3D Space," *Reverse Engineering, 2002. Proceedings. Ninth Working Conference,* Department of Computer Science, Concordia University, Montreal, Que., Canada, pp. 299-308.

[19] Swanson, G., and L. Globus (2001) "Software Metrics Visualization For a Graphical Programming Environment," *EE-Evaluation Engineering*, Nelson Publishing Inc.

[20] "Software Metric," *Wikipedia, the free encyclopedia,* Retrieved from http://en.wikipedia.org/wiki/Software_metric

[21] "Source Lines of Code," *Wikipedia, the free encyclopedia,* Retrieved from: http://en.wikipedia.org/wiki/Lines_of_code

[22] Tessier, J. (2001) Dependency Finder [Computer software], Retrieved from: http://depfind.sourceforge.net/

[23] "Visualization," *Merriam-Webster Online Dictionary,* Retrieved from: http://www.m-w.com/dictionary/visualization

APPENDICES

A.1. Survey Instructions

PROCEDURE

This Survey requires a person to participate in 2 different tests:

Raw Metric Test: Analyze the program from text file with provided data and answers the questions.

3D visualization Test. Analyze the program from 3D visualization program and answer the questions.

For Student at San Macros Campus please run Raw Metric test first and 3D visualization Test second

For Student at RRHCE Campus please run 3D visualization Test first and Raw Metric test second.

Instructions to run Raw Metric test

1. Go to MCS592 (San Marcos) or CS Open Lab (RRHCE) and logon to the Windows computer that
   labeled "3D Module Visualization Software survey installed".
2. Click on My Computer and click on C drive and locate the folder called **3DVisSurvey.**
3 Click on the Folder **"RawMetrics"** in the folder **3DVisSurvey.**
4. Open the text file within the folder
5 Complete the survey questions.

Instructions to run 3D visualization program test

1. Go to MCS592 (San Marcos) or CS Open Lab (RRHCE) and logon to the Windows computer that
   labeled "3D Module Visualization Software survey installed"
2 Click on **"Start"** under the menu bar
3. Click on **"Run"** within the Menu
4. Type in **"cmd"** to open the command prompt.
5 Type **"path=C: \Program Files\Java\jdk1.5.0_02\bin\"** to set the java3d home environment which allow user to run java commands.
6. Now switch your current directory to the folder **"C:\ 3DVisSurvey\3DVis"**
7. Type in **"java Model"** to run the program
   (It might take couple seconds for the program to be loaded and user might need to move or drag the java runtime window a bit for the 3D visualization to show up)
8. Complete the survey questions

3D visualization program control setting

1. Hold down the left mouse button and move around allows rotation of visualization system at the mouse point.
2 Hold down the right mouse button and move around allows translation of the visualization system at the mouse point.
3. Num pad '+' and '-' key allows zooming in and zooming out.
4. Metrics Configurations: In our visualization program, the setting of metric is defined within the file config.txt . User may change the setting in the config txt to fit his/her desired view of the software.

Each measurement in the configuration file is defined by an integer value (range between 0 - 4) which represents the different attributes of the module within the system

Integer 0 → Default
Integer 1 → Artifact Attribute 1
Integer 2 → Artifact Attribute 2
Integer 3 → Artifact Attribute 3
Integer 4 → Artifact Attribute 4

Notes on Software measurement

In software engineering, we use the modularity, the size, the complexity, the cohesion and the coupling to measure software systems. Cohesion and coupling are fundamental criteria of the understandability of a software system and also allow prediction of software quality. Well designed software usually has a high cohesion level and a low coupling level relation between modules within the system.

Legend used within the program

SLOC/LOC	=	lines of code
COH	=	cohesion
COU	=	coupling
RCM	=	relative complexity
CC	=	Cyclomatic complexity
PARAM	=	number of parameters within the module

A.2. Survey Test Questions

For any question or problem of how to get the program to work please email tk56539@txstate.edu

SURVEY PARTICPANT NAME: _____

Survey Questions (give an estimate time of how long you took to answer the following questions, each test shouldn't take an user more than 15 minutes, use your best judgment)

<<Starting Time>> _____ (round to the nearest second)

1 Where are you taking this survey (RRHEC or San Marcos)?
Ans:

2. How many total files (layers) are there in this program?
Ans:

3. How many total modules (artifacts) exist in this program?
Ans:

4 List all the metrics that were used to describe the system?
Ans:

5. Indicate which module has the highest lines of code in this program?
Ans:

6a. (This applies to Program A only)
List two modules which have the most relative complexity in the program?
Ans:

6b (This applies to Program B only)
List two modules which have the highest cohesion level in the program?
Ans:

7a. (This applies to Program A only)
List two modules which contain the most operators?
Ans:

7b. (This applies to Program B only)
List two modules which have the highest coupling level in the program?
Ans:

8. Which module is the most likely to be faulty?
(A faulty module most likely to have high coupling, low cohesion, high relative complexity(RCM), and high cyclomatic complexity(CC))

Ans:

9. Which module would be most likely to affect the entire system if modified?
Ans:

10a (This applies to Program A only)
A programmer is thinking about modifying the code and adding some features in the module "chk_token" is that a good idea? Why or Why not?
Ans:

10b. (This applies to Program B only)
A programmer is thinking about modifying the code and adding some features in the module "Shooter3DBehaviour.pickResultInfo" is that a good idea? Why or Why not?
Ans:

<<Finishing Time >> _____ (round to the nearest second)
Total time spends to take the Test A:
Total time spends to take the Test B:

```
//artifactclass.java
public class artifactclass
{
	private int id;
	private String name;
	private double attribute1;
	private double attribute2;
	private double attribute3;
	private double attribute4;
	private double attribute5;
	private float artifact_x;
	private float artifact_y;
	private float artifact_z;
	private int layer;
	private int[] linkid=new int[10];
	private int[] uplinkid=new int[10];
	private int numoflinks;
	private int numofuplinks;

	public artifactclass(int temp_id, String temp_name, double
temp_attribute1, double temp_attribute2, double temp_attribute3, double
temp_attribute4,double temp_attribute5,int temp_layer,int link_index,
int temp_linkid,int uplink_index,int temp_uplinkid)
	{
		id=temp_id;
		name=temp_name;
		attribute1=temp_attribute1;
		attribute2=temp_attribute2;
		attribute3=temp_attribute3;
		attribute4=temp_attribute4;
		attribute5=temp_attribute5;
		layer=temp_layer;
		linkid[link_index]=temp_linkid;
		uplinkid[uplink_index]=temp_uplinkid;
		numoflinks=0;
		numofuplinks=0;
		artifact_x=0.0f;
		artifact_y=0.0f;
		artifact_z=0.0f;
	}

	//set
	public void setid(int temp_id)
	{
		id=temp_id;
```

```
}
public void setattribute2(double temp_attribute2)
{
      attribute2=temp_attribute2;
}
public void setattribute3(double temp_attribute3)
{
      attribute3=temp_attribute3;
}
public void setattribute1(double temp_attribute1)
{
      attribute1=temp_attribute1;
}
public void setattribute4(double temp_attribute4)
{
      attribute4=temp_attribute4;
}
public void setattribute5(double temp_attribute5)
{
      attribute5=temp_attribute5;
}

public void setartifact_x(float temp_artifact_x)
{
      artifact_x=temp_artifact_x;
}
public void setartifact_y(float temp_artifact_y)
{
      artifact_y=temp_artifact_y;
}
public void setartifact_z(float temp_artifact_z)
{
      artifact_z=temp_artifact_z;
}

public void setname(String temp_name)
{
      name = temp_name;
}
public void setlayer(int temp_layer)
{
      layer = temp_layer;
}
public void setlinkid(int link_index, int temp_linkid)
{
      linkid[link_index] = temp_linkid;
}
public void setuplinkid(int uplink_index, int temp_uplinkid)
{
      uplinkid[uplink_index] = temp_uplinkid;
}
public void setnumoflinks(int temp_numoflinks)
{
      numoflinks = temp_numoflinks;
}
public void setnumofuplinks(int temp_numofuplinks)
{
```

```
        numofuplinks = temp_numofuplinks;
}

//get
public int getid()
{
        return id;
}
public String getname()
{
        return name;
}

public double getattribute1()
{
        return attribute1;
}

public double getattribute2()
{
        return attribute2;
}

public double getattribute3()
{
        return attribute3;
}

public double getattribute4()
{
        return attribute4;
}
public double getattribute5()
{
        return attribute5;
}

public float getartifact_x()
{
        return artifact_x;
}
public float getartifact_y()
{
        return artifact_y;
}
public float getartifact_z()
{
        return artifact_z;
}

public int getlayer()
{
        return layer;
}

public int getlinkid(int link_index)
{
```

```
		return linkid[link_index];
	}
	public int getuplinkid(int uplink_index)
	{
		return uplinkid[uplink_index];
	}
	public int getnumlinks()
	{
		return numoflinks;
	}
	public int getnumuplinks()
	{
		return numofuplinks;
	}

	public char printlinks()
	{
		for(int i=0;i<numoflinks;i++)
		{
			System.out.print(linkid[i]+" ");

		}
		//for(int i=0;i<linkid.length;i++)
		//{
		//	System.out.println(linkid[i]);
//		  }

		return ' ';
	}
	public char printuplinks()
	{
		for(int i=0;i<numofuplinks;i++)
		{
			System.out.print(uplinkid[i]+" ");

		}
		//for(int i=0;i<linkid.length;i++)
		//{
		//	System.out.println(linkid[i]);
		//	  }

		return ' ';
	}
}
```

```
//CreateParse.java
import java.io.File;
import java.io.InputStream;

import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.XMLReader;


public class CreateParser
{

	private DefaultHandler handler;
	private SAXParser saxParser;

	//constructor
	public CreateParser(DefaultHandler handler)
	{
		this.handler=handler;
		create();
	}

	//create the sax parser
	private void create()
	{
		try
		{
			SAXParserFactory factory =
SAXParserFactory.newInstance();
			factory.setNamespaceAware(true);
			factory.setValidating(true);
			saxParser = factory.newSAXParser();
		}
		catch (Throwable t)
		{
			t.printStackTrace();
		}
	}

	//parse a file

	public void parse(File file)
	{
		try
		{
			saxParser.parse(file,handler);
		}
		catch (Throwable t)
		{
			t.printStackTrace();
		}
	}

	//parse a URI
	public void parse(String uri)
```

```
        {
                try
                {
                        saxParser.parse(uri,handler);
                }
                catch (Throwable t)
                {
                        t.printStackTrace();
                }
        }

        //parse a stream
        public void parse(InputStream stream)
        {
                try
                {
                        saxParser.parse(stream,handler);
                }
                catch (Throwable t)
                {
                        t.printStackTrace();
                }
        }
}
```

```
//layerclass.java
public class layerclass
{
	private int id;
	private String name;
	private int numofartifacts;

	public layerclass(int temp_id, String temp_name)
	{
		id=temp_id;
		name=temp_name;
		numofartifacts=0;
	}

	//set
	public void setid(int temp_id)
	{
		id=temp_id;
	}

	public void setname(String temp_name)
	{
		name = temp_name;
	}

	public void setnumofartifacts(int temp_numofartifacts)
	{
		numofartifacts = temp_numofartifacts;
	}

	//get
	public int getid()
	{
		return id;
	}
	public String getname()
	{
		return name;
	}

	public int getnumofartifacts()
	{
		return numofartifacts;
	}

}
```

```
//OrderHandler.java
import java.text.NumberFormat;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.Attributes;


public class OrderHandler extends DefaultHandler
{
      int i=0,l=0;
      //private float fOrderPrice = 0;
      //private String priceElement="";
      private double lineofcode=0;
      double cohmin=0;
      double cohmax=1;
      double coumin=0;
      double coumax=1;
      double colmin=0;
      double colmax=1;

      public static String Program_name;
      static artifactclass[] artifactarray = new artifactclass[100];
      static layerclass[] layerarray = new layerclass[100];

      //entities
      private final String LAYER = "Layer";
      private final String LAYERS = "Layers";
      private final String ARTIFACT = "Artifact";
      private final String PROGRAM = "Program";
      private final String METRICS = "Metrics";


      //Program Attributes
      private final static String PROGRAMNAME = "3D Software Metrics
Visualization Engine";
      private final String GOODCOLOR = "GoodColor";
      private final String BADCOLOR = "BadColor";
      private final String TRANCOLOR = "TransitionColor";

      //Metrics Attributes
      private final String COHESIONMIN = "CohesionMin";
      private final String COHESIONMAX = "CohesionMax";
      private final String COUPLINGMIN = "CouplingMin";
      private final String COUPLINGMAX = "CouplingMax";
      private final String COLEMANMIN = "ColemanMaintainabilityMin";
      private final String COLEMANMAX = "ColemanMaintainabilityMax";

      //Artifact Attributes
      private final String UID = "UID";
      private final static String ATT1 = "COH";
      private final static String ATT2=  "COU";
      private final static String ATT3 = "SLOC";
      private final static String ATT4 = "PARAM";
      private final static String ATT5 = "DEFAULT";
      private final String LABEL ="Label";
      private final String LINKS = "Links";
```

```
private final String LID = "LID";
private final String ULID = "ULID";

public static int numberArtifacts=0;
public static int numberLayers = 0;
private int temp_lid_index=0;
private int temp_ulid_index=0;

//FLAGS
private boolean uid_flag=false;
private boolean att1_flag=false;
private boolean att2_flag=false;
private boolean att3_flag=false;
private boolean att4_flag=false;
private boolean att5_flag=false;

private boolean links_flag=false;
private boolean lid_flag=false;
private boolean ulid_flag=false;
private boolean label_flag=false;
private boolean artifact_flag=false;

private boolean program_flag=false;
private boolean programname_flag=false;
private boolean metrics_flag=false;
private boolean cohesionmin_flag=false;
private boolean cohesionmax_flag=false;
private boolean couplingmin_flag=false;
private boolean couplingmax_flag=false;
private boolean colemanmin_flag=false;
private boolean colemanmax_flag=false;

public static String getatt1name()
{
      return ATT1;
}

public static String getatt2name()
{
      return ATT2;
}

public static String getatt3name()
{
      return ATT3;
}

public static String getatt4name()
{
      return ATT4;
}

public static String getatt5name()
{
      return ATT5;
}
```

```
      public void startElement(String namespaceURL, String localName,
String qName, Attributes atts) throws SAXException
      {
            if (LAYER.equals(localName))
            {
                  numberLayers++;
                  layerclass layer=new layerclass(0,"layertesting");
                  layerarray[l]=layer;
            }
            if (ARTIFACT.equals(localName))
            {
                  numberArtifacts++;
                  artifactclass artifact =new
artifactclass(0,"testing",0.0,0.0,0.0,0.0,0.0,numberLayers,0,0,0,0);
                  artifactarray[i]=artifact;
                  artifact_flag=true;
            }
            if (LABEL.equals(localName))
            {
                  label_flag=true;
            }
            if (UID.equals(localName))
            {
                  uid_flag=true;
            }
            if (ATT1.equals(localName))
            {
                  att1_flag=true;
            }
            if (ATT2.equals(localName))
            {
                  att2_flag=true;
            }
            if (ATT3.equals(localName))
            {
                  att3_flag=true;
            }
            if (ATT4.equals(localName))
            {
                  att4_flag=true;
            }
            if (ATT5.equals(localName))
            {
                  att5_flag=true;
            }
            if (LINKS.equals(localName))
            {
                  links_flag=true;
            }
            if (LID.equals(localName))
            {
                  lid_flag=true;
            }
            if (ULID.equals(localName))
            {
                  ulid_flag=true;
            }
```

```
            if (PROGRAM.equals(localName))
            {
                  program_flag=true;
            }
            if (PROGRAMNAME.equals(localName))
            {
                  programname_flag=true;
            }

            if (METRICS.equals(localName))
            {
                  metrics_flag=true;
            }
            if (COHESIONMIN.equals(localName))
            {
                  cohesionmin_flag=true;
            }
            if (COHESIONMAX.equals(localName))
            {
                  cohesionmax_flag=true;
            }
            if (COUPLINGMIN.equals(localName))
            {
                  couplingmin_flag=true;
            }
            if (COUPLINGMAX.equals(localName))
            {
                  couplingmax_flag=true;
            }

            if (COLEMANMIN.equals(localName))
            {
                  colemanmin_flag=true;
            }
            if (COLEMANMAX.equals(localName))
            {
                  colemanmax_flag=true;
            }




      }

      public void characters(char[] ch, int start, int length) throws
SAXException
      {
            double lineofcode=0;
            int uid=0;
            int lid=0;
            int ulid=0;
            double att1=0;
            double att2=0;
            double att3=0;
            double att4=0;
```

```
                double att5=0;

                String metricData = (new String(ch, start, length)).trim();

                if(programname_flag==true)
                {
                      Program_name = new String(ch,start,length);
                }

                if(cohesionmin_flag==true)
                {
                      cohmin = Double.parseDouble(metricData);
                      cohesionmin_flag=false;
                }

                if(cohesionmax_flag==true)
                {
                      cohmax = Double.parseDouble(metricData);
                      cohesionmax_flag=false;
                }

                if(couplingmin_flag==true)
                {
                      coumin = Double.parseDouble(metricData);
                      couplingmin_flag=false;
                }

                if(couplingmax_flag==true)
                {
                      coumax = Double.parseDouble(metricData);
                      couplingmax_flag=false;
                }

                if(colemanmin_flag==true)
                {
                      colmin = Double.parseDouble(metricData);
                      colemanmin_flag=false;
                }

                if(colemanmax_flag==true)
                {
                      colmax = Double.parseDouble(metricData);
                      colemanmax_flag=false;
                }


                ////////////////This is parse label under
artifact////////////
                if(label_flag==true&&artifact_flag==true)
                {
                      String strValue = new String(ch,start,length);
                      try
                      {
                            //System.out.println(strValue);
                            artifactarray[i].setname(strValue);
                            label_flag=false;
                            artifact_flag=false;
```

```
                }
                catch(Exception e)
                {
                        //System.out.println("Can't parse the element -
" +e);
                }
        }

        if(label_flag==true&&artifact_flag==false)
        {
                String strValue = new String(ch,start,length);
                try
                {
                        //System.out.println(strValue);
                        layerarray[l].setname(strValue);
                        label_flag=false;
                }
                catch(Exception e)
                {
                        //System.out.println("Can't parse the element -
" +e);
                }
        }

        String chData = (new String(ch, start, length)).trim();
        if(chData.indexOf("\n") < 0 && chData.length() > 0)
        {
                if(uid_flag)
                {
                        uid = Integer.parseInt(chData);
                        artifactarray[i].setid(uid);
                        uid_flag=false;
                }

                if(att2_flag)
                {
                        att2 = Double.parseDouble(chData);
                        artifactarray[i].setattribute2(att2);
                        att2_flag=false;
                }
                if(att3_flag)
                {
                        att3 = Double.parseDouble(chData);
                        artifactarray[i].setattribute3(att3);
                        att3_flag=false;
                }
                if(att4_flag)
                {
                        att4 = Double.parseDouble(chData);
                        artifactarray[i].setattribute4(att4);
                        att4_flag=false;
                }
                if(att5_flag)
                {
                        att5 = Double.parseDouble(chData);
                        artifactarray[i].setattribute5(att5);
                        att5_flag=false;
```

```
                }
                if(att1_flag)
                {
                        att1 = Double.parseDouble(chData);
                        artifactarray[i].setattribute1(att1);
                        att1_flag=false;
                }
                if(links_flag)
                {
                        if(lid_flag)
                        {
                                lid = Integer.parseInt(chData);
                                //System.out.println("lid is now "+lid);

        artifactarray[i].setlinkid(temp_lid_index,lid);
                                temp_lid_index++;

        artifactarray[i].setnumoflinks(temp_lid_index);
                                lid_flag=false;
                        }
                        if(ulid_flag)
                        {
                                ulid = Integer.parseInt(chData);
                                //System.out.println("lid is now "+lid);

        artifactarray[i].setuplinkid(temp_ulid_index,ulid);
                                temp_ulid_index++;

        artifactarray[i].setnumofuplinks(temp_ulid_index);
                                ulid_flag=false;
                        }
                }

                }
        }

        public void endElement(String namespaceURI, String localName,
String qName) throws SAXException
        {
                if(ARTIFACT.equals(localName))
                {
                        //System.out.println(Program_name);
                        System.out.println("artifact"+i+" id:
"+artifactarray[i].getid());
                        System.out.println("artifact"+i+" name:
"+artifactarray[i].getname());
                        System.out.println("artifact"+i+" attribute 1:
"+artifactarray[i].getattribute1());
                        System.out.println("artifact"+i+" attribute 2:
"+artifactarray[i].getattribute2());
                        System.out.println("artifact"+i+" attribute 3:
"+artifactarray[i].getattribute3());
                        System.out.println("artifact"+i+" attribute 4:
"+artifactarray[i].getattribute4());
                        System.out.println("artifact"+i+" attribute 5:
"+artifactarray[i].getattribute5());
                        System.out.println("artifact"+i+" links:");
```

```
                    System.out.println(artifactarray[i].printlinks());
                    System.out.println(artifactarray[i].printuplinks());
                    System.out.println("artifact"+i+" locates at
layer:"+artifactarray[i].getlayer()+" "+layerarray[l].getname());
                    i++;

                    System.out.println("\nThe number of artifacts are
now:" + numberArtifacts +"\n");
              }
              else if(LAYER.equals(localName))
              {
                    System.out.println("\nThe number of layer are now:" +
numberLayers +"\n");
                    l++;

              }
              else if(LINKS.equals(localName))
              {
                    temp_lid_index=0;
                    links_flag=false;
              }
              else if(METRICS.equals(localName))
              {
                    System.out.println("cohmin is " + cohmin);
                    System.out.println("cohmax is "+ cohmax+"\n");
                    System.out.println("coumin is " + coumin);
                    System.out.println("coumax is "+ coumax+"\n");
                    System.out.println("colmin is " + colmin);
                    System.out.println("colmax is "+ colmax+"\n");
                    metrics_flag=false;
              }

              else if(PROGRAMNAME.equals(localName))
              {
                    System.out.println("\n\n" +Program_name+"\n\n");
                    program_flag=false;
              }


        }
        public static String getsysname()
        {
              return PROGRAMNAME;
              //return Program_name;
        }


}
```

```
//SAXSample.java
public class SAXSample
{
      public static void main(String[] args)
      {
            SAXSample jfs = new SAXSample();

            //Create Order's Handler
            OrderHandler oHandler = new OrderHandler();

            //Create the parser
            CreateParser parser = new CreateParser(oHandler);

            // Parse the XML file, handler generates the output
            parser.parse("test1.xml");

            //System.out.println("\n\n The Order.xml parsed - found
ORders: "+ oHandler.getNumberOrders());
      }
}
```

```
//VisEngine.java
import java.io.*;
import java.applet.Applet;
import java.awt.FlowLayout;
import java.awt.*;
import java.awt.event.*;
import java.awt.GraphicsConfiguration;
import com.sun.j3d.utils.applet.MainFrame;
import com.sun.j3d.utils.geometry.*;
import com.sun.j3d.utils.geometry.Box;
import javax.media.j3d.GeometryArray;
import com.sun.j3d.utils.universe.*;
import javax.media.j3d.*;
import javax.media.j3d.Appearance;
import javax.vecmath.*;
import javax.swing.*;
import com.sun.j3d.utils.behaviors.mouse.*;
import com.sun.j3d.utils.behaviors.keyboard.*;
import java.awt.Font;
import java.awt.event.MouseListener;
import com.sun.j3d.utils.behaviors.picking.*;
import java.util.Enumeration;
import java.util.StringTokenizer;
import com.sun.j3d.utils.picking.*;
import com.sun.j3d.utils.picking.behaviors.*;
import java.util.*;
import java.io.*;



public class VisEngine extends Applet
//public class Model extends MouseAdapter
{
    Canvas3D s1 = new
Canvas3D(SimpleUniverse.getPreferredConfiguration());

    static int numlinecode;
    static int numlayer;
      int temp_layerlocation;
    static MainFrame mf;
    private SimpleUniverse u = null;
    private BranchGroup scene = null;
      static int [] int_array= new int[7];      //array of setting from
config file

      private PickCanvas pickCanvas;

      ////////////////////////////setting the scene
up//////////////////////////////////////////
    public void init()
    {

            //setLayout(new FlowLayout());
            GraphicsConfiguration
config=SimpleUniverse.getPreferredConfiguration();
```

```
            s1.setSize(1024, 768);
            add(s1);

            ////////////////////////////////reading from
config////////////////////////////////
            final int MAX=4;
            StringTokenizer tokenizer;
            String line, name, file="config.txt";
            try
            {
                  FileReader fr = new FileReader(file);
                  BufferedReader inFile=new BufferedReader(fr);
                  line = inFile.readLine();
                  int i=0;
                  while(line != null)
                  {
                        tokenizer = new StringTokenizer(line);
                        name = tokenizer.nextToken();
                        try
                        {

      int_array[i]=Integer.parseInt(tokenizer.nextToken());
                        }
                        catch(NumberFormatException exception)
                        {
                              System.out.println("ERROR in put. Line
ignored:");
                              System.out.println(line);
                        }
                        line=inFile.readLine();
                        i++;
                  }
                  inFile.close();
            }
            catch (FileNotFoundException exception)
            {
                  System.out.println("The file "+file+" was not
found.");
            }
            catch(IOException exception)
            {
                  System.out.println(exception);
            }

      ////////////////////////////////////////////////////////////////
//////////////////////


            ////////////////// Create a simple scene and attach it to
the virtual universe//////////////
        scene = createSceneGraph(s1);
        u = new SimpleUniverse(s1);
        u.getViewingPlatform().setNominalViewingTransform();
        u.addBranchGraph(scene);
      }
```

```
    public BranchGroup createSceneGraph(Canvas3D s1)
    {

        ////////////////////// Create the root of the branch
graph////////////////////////////////
        BranchGroup objRoot = new BranchGroup();
        TransformGroup objTrans = new TransformGroup();

    objTrans.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

    objTrans.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);

        ////////////////////// Create a light that shines for 500m
from the origin/////////////////
        Color3f light1Color = new Color3f(1.5f, 1.5f, 1.8f);
        BoundingSphere bounds =  new BoundingSphere(new
Point3d(0.0, 0.0, 0.0), 500.0);
        Vector3f light1Direction = new Vector3f(4.0f, -7.0f, -
12.0f);
        DirectionalLight light1 = new DirectionalLight(light1Color,
light1Direction);
        light1.setInfluencingBounds(bounds);
        objTrans.addChild(light1);

        ////////////////////// Set
Colors/////////////////////////////////////////////////
        Color3f black = new Color3f(0.0f, 0.0f, 0.0f);
        Color3f white = new Color3f(1.0f, 1.0f, 1.0f);
        Color3f grey = new Color3f(0.3f,0.1f,0.1f);
        Color3f red = new Color3f(0.6f, .15f, .15f);
        Color3f green = new Color3f(0.15f, .6f, .15f);
        Color3f blue  = new Color3f(0.15f, .15f, .6f);
        Color3f lightblue  = new Color3f(0.3f, .4f, .6f);

    ////////////////////////////////////////////////////////////////
//////////////////

        ////////////////////////////////seting data from
config//////////////////////////
        int x_flag=int_array[0];
        int y_flag=int_array[1];
        int size_flag=int_array[2];
        int hue_flag=int_array[3];
        int intensity_flag=int_array[4];
        int   relationship_flag=int_array[5];
        int helperregions_flag=int_array[6];

        int color_flag=5;
        boolean test_flag=false;
        float good_xmin=0.0f;
        float good_xmax=0.25f;
        float good_ymin=0.0f;
        float good_ymax=0.25f;
        float bad_xmin=0.0f;
        float bad_xmax=-0.25f;
        float bad_ymin=0.0f;
        float bad_ymax=-0.25f;
```

```
            float good_xdelta=good_xmax-good_xmin;
            float good_ydelta=good_ymax-good_ymin;
            float bad_xdelta=bad_xmax-bad_xmin;
            float bad_ydelta=bad_ymax-bad_ymin;

      ////////////////////////////////////////////////////////////////
//////////////////

            float temp_z=0.0f;
            float temp_x=0.0f;
            float temp_y=0.0f;
            float value;
            int layer_index,artifact_index;
            int temp_numberartifacts=0;
            int temp_k;

            Point3f initPosition = new Point3f( 0.0f, 0.0f, 0.0f );
            Transform3D t = new Transform3D();
            t.transform(initPosition);
            TransformGroup tg = new TransformGroup(t);
            tg.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);
            tg.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

            temp_numberartifacts=OrderHandler.numberArtifacts;

            ////////////////////////// Creating text
labels//////////////////////////////////
            Font3D font3d = new Font3D(new
Font("Helvetica",Font.PLAIN,1), new FontExtrusion());

            ////////////////////////Program
Label//////////////////////////////
            Text3D text_title = new Text3D(font3d, new
String(OrderHandler.getsysname()), new Point3f(-25f,29f,temp_z));
            Text3D text_title1 = new Text3D(font3d, new String("
Artifact Color Hue Setting:         Blue = High    Yellow = Low"), new
Point3f(0f,27f,temp_z));
            Text3D text_title2 = new Text3D(font3d, new String("
Artifact Color Intensity Setting:  Bright = High  Dim = Low"), new
Point3f(0f,26f,temp_z));
            Text3D text_title3 = new Text3D(font3d, new String("  "),
new Point3f(0f,-25f,temp_z));
            Text3D text_art_size= new Text3D(font3d, new String("N/A
"), new Point3f(0f,25f,temp_z));


            System.out.print("\n\n"+"------------------------------
"+"\n\n");


            for(layer_index=0;layer_index<numlayer;layer_index++)
            {
                  artifact_index=0;
                  temp_k=layer_index+1;
                  Transform3D layer_t = new Transform3D();
                  layer_t.setTranslation(new
Vector3f(0.0f,0.0f,temp_z));
```

```
                    TransformGroup layer_tg = new
TransformGroup(layer_t);

      layer_tg.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);

      layer_tg.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);


                    ////////////////////////x axis
Label////////////////////////////
                    Text3D text_x = new Text3D(font3d, new String("x-
axis"), new Point3f(-25f,-29f,temp_z));
                    if(x_flag==1)
                    {
                          text_x.setString(new
String(OrderHandler.getatt1name()+"  >>>>>"));
                    }
                    else if(x_flag==2)
                    {
                          text_x.setString(new
String(OrderHandler.getatt2name()+"  >>>>>"));
                    }
                    else if(x_flag==3)
                    {
                          text_x.setString(new
String(OrderHandler.getatt3name()+"  >>>>>"));
                    }
                    else if(x_flag==4)
                    {
                          text_x.setString(new
String(OrderHandler.getatt4name()+"  >>>>>"));
                    }
                    else if(x_flag==5)
                    {
                          text_x.setString(new
String(OrderHandler.getatt5name()+"  >>>>>"));
                    }


                    Shape3D textShape1 = new Shape3D(text_x);
                    Transform3D text_t =new Transform3D();
                    text_t.set(0.02f);
                    TransformGroup text_tg = new TransformGroup();
                    text_tg.setTransform(text_t);

                    ////////////////////////////////y axis
Label//////////////////////////////////////////
                    Text3D text_y = new Text3D(font3d, new String("y-
axis"), new Point3f(-26f,28f,temp_z));
                    if(y_flag==1)
                    {
                          text_y.setString(new
String(OrderHandler.getatt1name()+"  >>>>>"));
                    }
                    else if(y_flag==2)
                    {
```

```
                        text_y.setString(new
String(OrderHandler.getatt2name()+"  >>>>>"));
                    }
                    else if(y_flag==3)
                    {
                        text_y.setString(new
String(OrderHandler.getatt3name()+"  >>>>>"));
                    }
                    else if(y_flag==4)
                    {
                        text_y.setString(new
String(OrderHandler.getatt4name()+"  >>>>>"));
                    }
                    else if(y_flag==5)
                    {
                        text_y.setString(new
String(OrderHandler.getatt5name()+"  >>>>>"));
                    }

                    Shape3D textShape2 = new Shape3D(text_y);
                    Transform3D text_t1 =new Transform3D();
                    text_t1.set(0.02f);
                    TransformGroup text_tg1 = new TransformGroup();
                    text_tg1.setTransform(text_t1);
                    Transform3D text_t2 =new Transform3D();
                    text_t2.rotZ(1.55f);
                    TransformGroup text_tg2 = new TransformGroup();
                    text_tg2.setTransform(text_t2);

                    text_tg.addChild(textShape1);
                    text_tg1.addChild(textShape2);
                    text_tg2.addChild(text_tg1);

      ////////////////////////////////////////////////////////////////
////////////////////


                    ////////////////////////////layer
labels////////////////////////////
                    Text3D text_layer = new Text3D(font3d, new
String(OrderHandler.layerarray[layer_index].getname()), new Point3f(-
10f,13f,temp_z));
                    Shape3D textShape3 = new Shape3D(text_layer);
                    Transform3D text_t3 =new Transform3D();
                    text_t3.set(0.04f);
                    TransformGroup text_tg3 = new TransformGroup();
                    text_tg3.setTransform(text_t3);
                    text_tg3.addChild(textShape3);

                    layer_tg.addChild(text_tg);
                    layer_tg.addChild(text_tg2);
                    layer_tg.addChild(text_tg3);


      ////////////////////////////////////////////////////////////////
////////////////////
```

```
                    ////////////////////////////// Creating our layers
////////////////////////////////////
                    Appearance app_layer = new Appearance();
                    TransparencyAttributes ta = new
TransparencyAttributes( );
                    ta.setTransparency( 0.5f );
                    ta.setTransparencyMode(
TransparencyAttributes.BLENDED );
                    app_layer.setMaterial(new Material(lightblue, black,
lightblue, black, 1.0f));
                    app_layer.setTransparencyAttributes( ta );

                    Box box = new Box(0.5f,0.5f,0.005f,null);
                    box.setAppearance(app_layer);

                    layer_tg.addChild(box);

                    ////////////////////////////creating test layer if
turned ON from config//////////////////
                    Transform3D testlayergood_t = new Transform3D();
                    testlayergood_t.setTranslation(new
Vector3f(good_xmin+good_xdelta,good_ymin+good_ydelta,temp_z));

                    TransformGroup testlayergood_tg = new
TransformGroup(testlayergood_t);

      testlayergood_tg.setCapability(TransformGroup.ALLOW_TRANSFORM_REA
D);

      testlayergood_tg.setCapability(TransformGroup.ALLOW_TRANSFORM_WRI
TE);

                    Appearance app_testlayergood = new Appearance();
                    TransparencyAttributes testgood_ta = new
TransparencyAttributes(TransparencyAttributes.BLENDED,      0.2f
,TransparencyAttributes.BLEND_SRC_ALPHA,TransparencyAttributes.BLEND_ON
E);
                    app_testlayergood.setMaterial(new Material(green,
black, green, black, 1.0f));
                    app_testlayergood.setTransparencyAttributes(
testgood_ta );


                    Transform3D testlayerbad_t = new Transform3D();
                    testlayerbad_t.setTranslation(new
Vector3f(bad_xmin+bad_xdelta,bad_ymin+bad_ydelta,temp_z));

                    TransformGroup testlayerbad_tg = new
TransformGroup(testlayerbad_t);

      testlayerbad_tg.setCapability(TransformGroup.ALLOW_TRANSFORM_READ
);

      testlayerbad_tg.setCapability(TransformGroup.ALLOW_TRANSFORM_WRIT
E);

                    Appearance app_testlayerbad = new Appearance();
```

```
                    TransparencyAttributes testbad_ta = new
TransparencyAttributes(TransparencyAttributes.BLENDED,      0.2f
,TransparencyAttributes.BLEND_SRC_ALPHA,TransparencyAttributes.BLEND_ON
E);
                    app_testlayerbad.setMaterial(new Material(red, black,
red, black, 1.0f));
                    app_testlayerbad.setTransparencyAttributes(
testbad_ta );

                    if(helperregions_flag!=0)
                    {
                          Box testgood_box = new
Box(good_xdelta,good_ydelta,0.008f,null);
                          Box testbad_box = new
Box(bad_xdelta,bad_ydelta,0.008f,null);

                          testgood_box.setAppearance(app_testlayergood);
                          testbad_box.setAppearance(app_testlayerbad);

                          testlayergood_tg.addChild(testgood_box);
                          testlayerbad_tg.addChild(testbad_box);
                    }

      ////////////////////////////////////////////////////////////////
//////////////////////


                    while(temp_numberartifacts>0)
                    {

      //temp_layerlocation=OrderHandler.artifactarray[artifact_index].g
etlayer();
                          //System.out.println("temp_layerlocation is
"+temp_layerlocation);

                          /////testing whether the artifact's location
and the current layer location match, if YES draw)////

      if(OrderHandler.artifactarray[artifact_index].getlayer() ==
temp_k)
                          {
                                //System.out.println("attribute1 is
"+OrderHandler.artifactarray[artifact_index].getattribute2());
                                //System.out.println("attribute2 is
"+OrderHandler.artifactarray[artifact_index].getattribute2());
                                //System.out.println("attribute3 is
"+OrderHandler.artifactarray[artifact_index].getattribute3());
                                //System.out.println("attribute4 is
"+OrderHandler.artifactarray[artifact_index].getattribute2());
                                //System.out.println("attribute5 is
"+OrderHandler.artifactarray[artifact_index].getattribute2());

                                ////////////////////////setting the x
position of artifact from config////////////////////
                                if(x_flag==1)
                                {
```

```
      temp_x=((float)OrderHandler.artifactarray[artifact_index].getattr
ibute1()-0.5f);
                                }
                                else if(x_flag==2)
                                {

      temp_x=((float)OrderHandler.artifactarray[artifact_index].getattr
ibute2()-0.5f);
                                }
                                else if(x_flag==3)
                                {

      //temp_x=((float)OrderHandler.artifactarray[artifact_index].getat
tribute3()*0.001f-0.5f);

      temp_x=((float)OrderHandler.artifactarray[artifact_index].getattr
ibute3()-0.5f);
                                }
                                else if(x_flag==4)
                                {

      temp_x=((float)OrderHandler.artifactarray[artifact_index].getattr
ibute4()-0.5f);
                                }
                                else if(x_flag==5)
                                {

      temp_x=((float)OrderHandler.artifactarray[artifact_index].getattr
ibute5()-0.5f);
                                }
                                else
                                {

      temp_x=((float)OrderHandler.artifactarray[artifact_index].getattr
ibute1()-0.5f);
                                }

                                //////////////////////////setting the y
position of artifact from config////////////////////
                                if(y_flag==1)
                                {

      temp_y=((float)OrderHandler.artifactarray[artifact_index].getattr
ibute1()-0.5f);
                                }
                                else if(y_flag==2)
                                {

      temp_y=((float)OrderHandler.artifactarray[artifact_index].getattr
ibute2()-0.5f);
                                }
                                else if(y_flag==3)
                                {

      temp_y=((float)OrderHandler.artifactarray[artifact_index].getattr
ibute3()-0.5f);
```

```
      //temp_y=((float)OrderHandler.artifactarray[artifact_index].getat
tribute3()*0.001f-0.5f);
                                    }
                                    else if(y_flag==4)
                                    {

      temp_y=((float)OrderHandler.artifactarray[artifact_index].getattr
ibute4()-0.5f);
                                    }
                                    else if(y_flag==5)
                                    {

      temp_y=((float)OrderHandler.artifactarray[artifact_index].getattr
ibute5()-0.5f);
                                    }
                                    else
                                    {

      temp_y=((float)OrderHandler.artifactarray[artifact_index].getattr
ibute1()-0.5f);
                                    }

                                    //store the actual xyz postion in the
scene back to 3D objects for future use

      OrderHandler.artifactarray[artifact_index].setartifact_x(temp_x);

      OrderHandler.artifactarray[artifact_index].setartifact_y(temp_y);

      OrderHandler.artifactarray[artifact_index].setartifact_z(temp_z);


      System.out.println("artifact["+OrderHandler.artifactarray[artifac
t_index].getid()+"] x is:
"+OrderHandler.artifactarray[artifact_index].getartifact_x()+"\n");

      System.out.println("artifact["+OrderHandler.artifactarray[artifac
t_index].getid()+"] y is:
"+OrderHandler.artifactarray[artifact_index].getartifact_y()+"\n");

      System.out.println("artifact["+OrderHandler.artifactarray[artifac
t_index].getid()+"] z is:
"+OrderHandler.artifactarray[artifact_index].getartifact_z()+"\n");

                                    //System.out.println("temp_x is
"+temp_x);
                                    //System.out.println("temp_y is
"+temp_y);

                                    ////////////////////////////// Creating
our Artifacts(Spheres) ////////////////////////////
                                    Transform3D artifact_t = new
Transform3D();

                                    BoundingSphere behaveBounds = new
BoundingSphere();
```

```
                                artifact_t.setTranslation(new
Vector3f(temp_x,temp_y,temp_z));
                                TransformGroup artifact_tg = new
TransformGroup(artifact_t);

      artifact_tg.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);

      artifact_tg.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

      artifact_tg.setCapability(TransformGroup.ENABLE_PICK_REPORTING);

                                value=0;



                                ////////////////////////////setting the
size of sphere depends of the Config////////////////
                                if(size_flag==1)
                                {

      value=(float)OrderHandler.artifactarray[artifact_index].getattrib
ute1()*0.1f+0.01f;
                                      text_art_size.setString(new
String("Artifact size is reperesented as
"+OrderHandler.getatt1name()+":  Large=High  Small=Low"));
                                }
                                else if(size_flag==2)
                                {

      value=(float)OrderHandler.artifactarray[artifact_index].getattrib
ute2()*0.1f+0.01f;
                                      text_art_size.setString(new
String("Artifact size is reperesented as
"+OrderHandler.getatt2name()+":  Large=High  Small=Low"));
                                }
                                else if(size_flag==3)
                                {

      value=(float)OrderHandler.artifactarray[artifact_index].getattrib
ute3()*0.1f+0.01f;
                                      text_art_size.setString(new
String("Artifact size is reperesented as
"+OrderHandler.getatt3name()+":  Large=High  Small=Low"));
                                }
                                else if(size_flag==4)
                                {

      value=(float)OrderHandler.artifactarray[artifact_index].getattrib
ute4()*0.1f+0.01f;
                                      text_art_size.setString(new
String("Artifact size is reperesented as
"+OrderHandler.getatt4name()+":  Large=High  Small=Low"));
                                }
                                else if(size_flag==5)
                                {
```

```
      value=(float)OrderHandler.artifactarray[artifact_index].getattrib
ute5()*0.1f+0.01f;
                                    text_art_size.setString(new
String("Artifact size is reperesented as
"+OrderHandler.getatt5name()+":  Large=High  Small=Low"));
                                    }
                                    else
                                    {

      value=(float)OrderHandler.artifactarray[artifact_index].getattrib
ute1()*0.1f+0.01f;
                                    text_art_size.setString(new
String("Artifact size is reperesented as
"+OrderHandler.getatt1name()+":  Large=High  Small=Low"));
                                    }



                                    //System.out.println("Sphere actual
radius is " + value);

                                    Sphere sphere = new Sphere(value);
                                    sphere.getShape().setCapability (
Shape3D.ALLOW_APPEARANCE_READ );
                                    sphere.getShape().setCapability (
Shape3D.ALLOW_APPEARANCE_WRITE );
                                    Appearance ap = new Appearance();
                                    ap.setColoringAttributes (new
ColoringAttributes (new Color3f (0.0f, 0.0f, 0.0f),1));


                                    ////////////////////////////setting the
Color Hue of sphere depends of the Config/////////////////
                                    Color3f temp_color  = new Color3f(0.5f,
.5f, .5f);
                                    Color3f temp_intensity  = new
Color3f(0.0f, 0.0f, 0.0f);

                                    if(hue_flag==1)
                                    {
                                          temp_color=new Color3f(0.5f,
0.5f,(float)OrderHandler.artifactarray[artifact_index].getattribute1())
;
                                          text_title1.setString(new
String("Artifact Hue is reperesented as "+OrderHandler.getatt1name()+":
Blue=High  Yellow=Low"));
                                    }
                                    else if(hue_flag==2)
                                    {
                                          temp_color=new Color3f(0.5f,
0.5f,(float)OrderHandler.artifactarray[artifact_index].getattribute2())
;
                                          text_title1.setString(new
String("Artifact Hue is reperesented as "+OrderHandler.getatt2name()+":
Blue=High  Yellow=Low"));
                                    }
```

```
                                else if(hue_flag==3)
                                {
                                        temp_color=new Color3f(0.5f,
0.5f,(float)OrderHandler.artifactarray[artifact_index].getattribute3())
;
                                        text_title1.setString(new
String("Artifact Hue is reperesented as "+OrderHandler.getatt3name()+":
Blue=High  Yellow=Low"));
                                }
                                else if(hue_flag==4)
                                {
                                        temp_color=new Color3f(0.5f,
0.5f,(float)OrderHandler.artifactarray[artifact_index].getattribute4())
;
                                        text_title1.setString(new
String("Artifact Hue is reperesented as "+OrderHandler.getatt4name()+":
Blue=High  Yellow=Low"));
                                }
                                else if(hue_flag==5)
                                {
                                        temp_color=new Color3f(0.5f,
0.5f,(float)OrderHandler.artifactarray[artifact_index].getattribute5())
;
                                        text_title1.setString(new
String("Artifact Hue is reperesented as "+OrderHandler.getatt5name()+":
Blue=High  Yellow=Low"));
                                }
                                else
                                {
                                        temp_color=new Color3f(0.15f, 0.6f,
0.15f);
                                }


                                ////////////////////////////setting the
Color intensity of sphere depends of the Config////////////////
                                if(intensity_flag==1)
                                {
                                        temp_intensity=new
Color3f((float)OrderHandler.artifactarray[artifact_index].getattribute1
(),
(float)OrderHandler.artifactarray[artifact_index].getattribute1(),(floa
t)OrderHandler.artifactarray[artifact_index].getattribute1());
                                        text_title2.setString(new
String("Artifact Intensity is reperesented as
"+OrderHandler.getatt1name()+":  Bright=High  Dim=Low"));
                                }
                                else if(intensity_flag==2)
                                {
                                        temp_intensity=new
Color3f((float)OrderHandler.artifactarray[artifact_index].getattribute2
(),
(float)OrderHandler.artifactarray[artifact_index].getattribute2(),(floa
t)OrderHandler.artifactarray[artifact_index].getattribute2());
                                        text_title2.setString(new
String("Artifact Intensity is reperesented as
"+OrderHandler.getatt2name()+":  Bright=High  Dim=Low"));
```

```
                                }
                                else if(intensity_flag==3)
                                {
                                        temp_intensity=new
Color3f((float)OrderHandler.artifactarray[artifact_index].getattribute3
(),
(float)OrderHandler.artifactarray[artifact_index].getattribute3(),(floa
t)OrderHandler.artifactarray[artifact_index].getattribute3());
                                        text_title2.setString(new
String("Artifact Intensity is reperesented as
"+OrderHandler.getatt3name()+":  Bright=High  Dim=Low"));
                                }
                                else if(intensity_flag==4)
                                {
                                        temp_intensity=new
Color3f((float)OrderHandler.artifactarray[artifact_index].getattribute4
(),
(float)OrderHandler.artifactarray[artifact_index].getattribute4(),(floa
t)OrderHandler.artifactarray[artifact_index].getattribute4());
                                        text_title2.setString(new
String("Artifact Intensity is reperesented as
"+OrderHandler.getatt4name()+":  Bright=High  Dim=Low"));
                                }
                                else if(intensity_flag==5)
                                {
                                        temp_intensity=new
Color3f((float)OrderHandler.artifactarray[artifact_index].getattribute5
(),
(float)OrderHandler.artifactarray[artifact_index].getattribute5(),(floa
t)OrderHandler.artifactarray[artifact_index].getattribute5());
                                        text_title2.setString(new
String("Artifact Intensity is reperesented as
"+OrderHandler.getatt5name()+":  Bright=High  Dim=Low"));
                                }
                                else
                                {
                                        text_title2.setString(new
String("Artifact Intensity is reperesented as
"+OrderHandler.getatt5name()+":  Bright=High  Dim=Low"));
                                        temp_intensity=black;
                                }


                                ap.setMaterial(new Material(temp_color,
temp_intensity, temp_color, temp_intensity, 1.0f));
                                sphere.setAppearance(ap);

                                artifact_tg.addChild(sphere);
                                //artifact_tg.setPickable(true);
                                //PickRotateBehavior behavior =new
PickRotateBehavior(artifact_tg,s1,behaveBounds);
                                //artifact_tg.addChild(behavior);

                                ////////////////////////Labeling
Artifacts//////////////////////////////
                                float temp_xx=temp_x+1f;
                                float temp_yy=temp_y+1f;
```

```
                                float temp_zz=temp_z+1f+40*value;
                                Text3D text_artifact = new Text3D(font3d,
new String(OrderHandler.artifactarray[artifact_index].getname()), new
Point3f(temp_xx,temp_yy,temp_zz));
                                Shape3D textShape0 = new
Shape3D(text_artifact);
                                Transform3D text_t0 =new Transform3D();
                                text_t0.set(0.02f);
                                TransformGroup text_tg0 = new
TransformGroup();
                                text_tg0.setTransform(text_t0);
                                text_tg0.addChild(textShape0);
                                artifact_tg.addChild(text_tg0);

      ////////////////////////////////////////////////////////////////
//////////

                                tg.addChild(artifact_tg);

                          }
                          artifact_index++;
                          temp_numberartifacts--;
                    }



                    temp_z-=0.2;

                    tg.addChild(layer_tg);
                    tg.addChild(testlayergood_tg);
                    tg.addChild(testlayerbad_tg);
                    temp_numberartifacts=OrderHandler.numberArtifacts;


              }
              text_title3.setString(new String("Total number of layer(s):
"+OrderHandler.numberLayers+" / Total number of artifact(s):
"+OrderHandler.numberArtifacts));

              ////////////////////artifact legend labeler//////////////
              Shape3D textShapetitle = new Shape3D(text_title);
              Shape3D textShapetitle1 = new Shape3D(text_title1);
              Shape3D textShapetitle2 = new Shape3D(text_title2);
              Shape3D textShapetitle3 = new Shape3D(text_title3);

              Transform3D text_t01 =new Transform3D();
              text_t01.set(0.023f);
              TransformGroup text_tgtitle = new TransformGroup();
              text_tgtitle.setTransform(text_t01);
              text_tgtitle.addChild(textShapetitle);
              text_tgtitle.addChild(textShapetitle1);
              text_tgtitle.addChild(textShapetitle2);
              text_tgtitle.addChild(textShapetitle3);
              tg.addChild(text_tgtitle);

              Shape3D textShape4 = new Shape3D(text_art_size);
              Transform3D text_t4 =new Transform3D();
```

```
            text_t4.set(0.023f);
            TransformGroup text_tg4 = new TransformGroup();
            text_tg4.setTransform(text_t4);
            text_tg4.setTransform(text_t4);
            text_tg4.addChild(textShape4);

            tg.addChild(text_tg4);
            //    Frame frame = new Frame("Box and Sphere");
            //    frame.addWindowListener(new WindowAdapter()
            //    {
            //          public void windowClosing(WindowEvent winEvent)
            //          {
            //                System.exit(0);
            //          }
            //    });


            pickCanvas = new PickCanvas(s1,objRoot);
            pickCanvas.setMode(PickTool.GEOMETRY_INTERSECT_INFO );
            pickCanvas.setMode(PickCanvas.BOUNDS);
            s1.addMouseListener(new MyAdapter() );

            for (int v=0; v < objRoot.numChildren(); ++v)
            {

      ((Shape3D)(objRoot).getChild(v)).getGeometry().setCapability(Geom
etry.ALLOW_INTERSECT);

      //((Shape3D)(objRoot).getChild(v)).getGeometry().setCapability(AL
LOW_CHILDREN_READ);
                  //
      ((Shape3D)(objRoot).getChild(v)).getGeometry().setCapability(ALLO
W_CHILDREN_WRITE);
                  //System.out.print("testing123");
            }

            ////////////////////////////////////////// draw link
lines//////////////////////////
            int total_num_artifact_links=0;


            for(int temp_artifact_index=0;
temp_artifact_index<OrderHandler.numberArtifacts;temp_artifact_index++)
            {
                  //System.out.print("this one has links:
"+OrderHandler.artifactarray[temp_artifact_index].getnumlinks()+"\n");

      if(OrderHandler.artifactarray[temp_artifact_index].getnumlinks()!
=0)
                  {
                        for(int temp_link_index=0; temp_link_index <
OrderHandler.artifactarray[temp_artifact_index].getnumlinks() ;
temp_link_index++)
                              total_num_artifact_links++;
                  }

            }
```

```
            System.out.print("total_num_artifact_links
is"+total_num_artifact_links+"\n\n");



            int searh_uidtoindex=0;
            float temp_float=0.0f;
            int temp_artifact_index=0;
            int temp_link_index=0;
            float temp_lx;
            float temp_ly;
            float temp_lz;
            int temp_linkid;
            int vertexarray_index=0;

            float[] verts= new float[total_num_artifact_links*2*3];



            while(temp_artifact_index<OrderHandler.numberArtifacts)
            {
                  System.out.print("tempartifactindex is now:
"+temp_artifact_index+"\n");

                  temp_link_index=0;

                  while(temp_link_index <
OrderHandler.artifactarray[temp_artifact_index].getnumlinks())
                  {
                        System.out.print("templinkindex is now:
"+temp_link_index+"\n\n");
                        System.out.print("templinkindex is now:
"+temp_link_index+"\n");

      if(OrderHandler.artifactarray[temp_artifact_index].getnumlinks()!
=0)
                        {
                              System.out.print("the O_X is:
"+OrderHandler.artifactarray[temp_artifact_index].getartifact_x()+"\n")
;
                              System.out.print("the O_Y is:
"+OrderHandler.artifactarray[temp_artifact_index].getartifact_y()+"\n")
;
                              System.out.print("the O_Z is:
"+OrderHandler.artifactarray[temp_artifact_index].getartifact_z()+"\n\n
");


      verts[vertexarray_index]=OrderHandler.artifactarray[temp_artifact
_index].getartifact_x();
                              vertexarray_index++;

      verts[vertexarray_index]=OrderHandler.artifactarray[temp_artifact
_index].getartifact_y();
                              vertexarray_index++;
```

```
      verts[vertexarray_index]=OrderHandler.artifactarray[temp_artifact
_index].getartifact_z();
                              vertexarray_index++;


      temp_linkid=OrderHandler.artifactarray[temp_artifact_index].getli
nkid(temp_link_index);
                              System.out.print("the temp_linkid is :
"+temp_linkid+"\n");


      for(searh_uidtoindex=0;searh_uidtoindex<OrderHandler.numberArtifa
cts;searh_uidtoindex++)
                              {

      if(OrderHandler.artifactarray[searh_uidtoindex].getid()==temp_lin
kid)
                                    {
                                          break;
                                    }
                              }
                              System.out.print("the L_X is:
"+OrderHandler.artifactarray[searh_uidtoindex].getartifact_x()+"\n");

      verts[vertexarray_index]=OrderHandler.artifactarray[searh_uidtoin
dex].getartifact_x();
                              vertexarray_index++;
                              System.out.print("the L_Y is:
"+OrderHandler.artifactarray[searh_uidtoindex].getartifact_y()+"\n");

      verts[vertexarray_index]=OrderHandler.artifactarray[searh_uidtoin
dex].getartifact_y();
                              vertexarray_index++;
                              System.out.print("the L_Z is:
"+OrderHandler.artifactarray[searh_uidtoindex].getartifact_z()+"\n\n");

      verts[vertexarray_index]=OrderHandler.artifactarray[searh_uidtoin
dex].getartifact_z();
                              vertexarray_index++;

                              temp_link_index++;

                        }

                        //temp_float=temp_float+0.03f;
                        //temp_link_index++;
                  }
                  temp_artifact_index++;
            }


            for(int u=0; u<total_num_artifact_links*2*3;u++)
            {
                  if(u%3==0)
                  {
                        System.out.print("\n");
```

```
                        System.out.print(" "+verts[u]);

                }
                else
                {
                        System.out.print(" "+verts[u]);
                }

        }

        //QuadArray line;
        LineArray line;
        line = new LineArray(total_num_artifact_links*2,
GeometryArray.COORDINATES | GeometryArray.COLOR_3 );
        int counter001=0;

        for(int l=0;l<total_num_artifact_links*2;l++)
        {

                line.setCoordinate(l, new
Point3f(verts[counter001],verts[counter001+1],verts[counter001+2]));
                counter001=counter001+3;

                line.setCoordinate(l+1, new
Point3f(verts[counter001],verts[counter001+1],verts[counter001+2]));
                counter001=counter001+3;
                //    System.out.print("\nIam here" + l + "\n");

                line.setColor(l, new Color3f(0.02f,0.02f,1f));
                line.setColor(l+1, new Color3f(0.02f,1f,0.02f));

                System.out.print("\nIam here" + l + "\n");
                l++;

        }
        Shape3D s = new Shape3D(line);//shape contain the lines

        if(relationship_flag!=0)
        {
                tg.addChild(s);//insert the shape into branchGroup
        }

        objTrans.addChild(tg);



        ////////////////////////////////CREATING
BEHAVIORS////////////////////////////////////////

        ////////////////////////// create the rotate behavior
node//////////////////////////////////
        MouseRotate behavior_r = new MouseRotate();
        behavior_r.setTransformGroup(tg);
        objTrans.addChild(behavior_r);
```

```
            ////////////////////// Create the translate behavior
node//////////////////////////
            MouseTranslate behavior_t = new MouseTranslate();
            behavior_t.setTransformGroup(tg);
            objTrans.addChild(behavior_t);

            behavior_r.setSchedulingBounds(bounds);
            behavior_t.setSchedulingBounds(bounds);

            ////////////////////////setting keyboard
movement//////////////////////////////////
            KeyNavigatorBehavior keyNavBeh = new
KeyNavigatorBehavior(tg);
            keyNavBeh.setSchedulingBounds(new BoundingSphere(new
Point3d(),1000.0));
            objTrans.addChild(keyNavBeh);

            objRoot.addChild(objTrans);

        return objRoot;

    }

      public void destroy()
    {
        u.removeAllLocales();
    }

      public void menu()
      {
            System.out.print("MENU");
            System.out.print("1. Enter a Artifact name");
            System.out.print("2. Enter a Layer name");
            System.out.print("3. reset");
      }

      ////////////////////////////////////////////MAIN/////////////////
//////////////////////////
    public static void main(String[] args)
      {
            //Model jfs = new Model();

            //Create Order's Handler
            OrderHandler oHandler = new OrderHandler();

            //Create the parser
            CreateParser parser = new CreateParser(oHandler);

            // Parse the XML file, handler generates the output
            parser.parse("test1.xml");

            //getting total number of layers
            numlayer=OrderHandler.numberLayers;


            //new Model();
            mf = new MainFrame(new VisEngine(), 1024, 768);
```

```
    }


      class MyAdapter extends MouseAdapter
      {

                  public void mouseEntered(MouseEvent e)
                  {
                  }

                  public void mouseExited(MouseEvent e)
                  {
                  }

                  public void mousePressed(MouseEvent e)
                  {
                  }

                  public void mouseReleased(MouseEvent e)
                    {
                  }
              //public void processAWTEvent(MouseEvent e)
              public void mouseClicked(MouseEvent e)
              {
                    System.out.print("coordinates(" + e.getX() + ", " +
e.getY() + "): ");

                    Point3d eyePos = pickCanvas.getStartPosition(); //
get viewer eye location

                    pickCanvas.setShapeLocation(e.getX(),e.getY());//
register mouse pointer location on the screen

                    PickResult result = pickCanvas.pickClosest(); //get
the intersected shape closest to the viewer

                    if (result == null)
                    {
                          System.out.println("Nothing picked");
                    }
                    else
                    {

                          Shape3D s =
(Shape3D)result.getNode(PickResult.SHAPE3D);

                          System.out.println(s.getClass().getName());

      //s.getGeometry().setCapability(Geometry.ALLOW_INTERSECT);
                          //s.setCapability(Shape3D.ALLOW_GEOMETRY_READ);

      //s.setCapability(Shape3D.ALLOW_GEOMETRY_WRITE);
                          if(s != null)
                          {
                                System.out.print("MENU\n");
                                System.out.print("1. Enter a Artifact
name\n");
```

```
                                //System.out.print("2. Enter a Layer
name\n");
                                System.out.print("3. reset\n");

                                try
                                {
                                        InputStreamReader inStream = new
InputStreamReader( System.in ) ;
                                        BufferedReader stdin = new
BufferedReader( inStream );
                                        String inData;

                                        //inData = stdin.readLine();
                                        //while(inData!=null)
                                        //{

                                        //}
                                }
                                catch(NumberFormatException exception)
                                {
                                        System.out.println("Error in
input.Line ignored:");
                                        System.out.println("line");
                                }



                                //System.out.println(s.getClass() );
                                //PickIntersection
pi=result.getClosestIntersection(eyePos);
                                //Point3d
intercept=pi.getPointCoordinatesVW();
                            }

                            //}
                            /*
                             Primitive p =
(Primitive)result.getNode(PickResult.PRIMITIVE);

                             Shape3D s =
(Shape3D)result.getNode(PickResult.SHAPE3D);

                             TransformGroup  t =
(TransformGroup)result.getNode(PickResult.TRANSFORM_GROUP);

                             if (p != null)
                             {


System.out.println("hello");//p.getClass().getName());
                             //System.out.println(s.getGeometry() );


                             PickIntersection pi =
PickResult.getClosestIntersection(eyePos);
```

```
                        Point3d
intercept=pi.getClosestVertexCoordinatesVW();
                        }
                        if (s != null)
                        {
                        System.out.println(s.getClass().getName());
                        }
                        else
                        {
                        System.out.println("null");
                        }
                        */

                }

            }
        }

}
```

```
//TXT_XMLCONVERTER.CPP
#include <iostream>
#include <stdlib.h>
#include <fstream>
#include <string>
#include <math.h>
using namespace std;

void readfile(fstream& fin,char title[],char attrib[][50], float attrib_value[][50],string
name[],int& total_row,int& total_col,float min_value[],float max_value[]);
void normalize(float attrib_value[][50],int total_row,int total_col,float min_value[],float
max_value[]);
void printformat(fstream& fout,char title[],char attrib[][50], float attrib_value[][50],string
name[],int total_row,int total_col,int& artifact_id,float min_value[],float max_value[]);

struct layer
{
        char layer_name[50];
        char title[50];
        char attrib[50][50];
   string name[50];
        float attrib_value[100][50];
};

int main()
{
        int      artifact_id=0;
        float    min_value[50];
        float    max_value[50];

        //char layer_name;
   int i=0,j=0,k=1,row=0,col=0;
        int total_col, total_row;
        int total_layer;
        bool more_layer=0;
        layer layer[20];

        fstream fin;
        //fstream fin("test056.rcm",ios::in);
   fstream fout("input.xml",ios::out);
        ////////////////////////////////////////
   fout<<"<Layers>"<<endl;
        fout<<"\t<ProgramName>"<<"Sample System"<<"</ProgramName>"<<endl;
  for(int a;a<50;a++)
  {
                min_value[a]=99;
                max_value[a]=-99;
  }

  while(true)
  {
```

```
                cout<<"Enter a name for input layer"<<endl;
                cin>>layer[i].layer_name;
                fin.open(layer[i].layer_name);
                if(!fin)
                {
                        cout<<"Cannot locate the inputed layer
"<<layer[i].layer_name<<endl<<endl;
                        break;
                }

                readfile(fin,layer[i].title,layer[i].attrib,
layer[i].attrib_value,layer[i].name,total_row,total_col,min_value,max_value);

                for(j=0;j<50;j++)
                {
                        cout<<"min "<<j<<": "<<min_value[j]<<endl;
                        cout<<"max "<<j<<": "<<max_value[j]<<endl;
                }

                cout<<"\nDo you want to read another layer?"<<endl;
                cin>>more_layer;

                if(more_layer==0)
                {
                        break;
                        fin.close();
                }

                i++;
                fin.close();
                fin.clear();
        }
   total_layer=i;

        for(k=0;k<=total_layer;k++)
        {
                //cout<<"hello"<<endl;
                normalize(layer[k].attrib_value,total_row,total_col,min_value,max_value);
        printformat(fout,layer[k].title,layer[k].attrib,
layer[k].attrib_value,layer[k].name,total_row,total_col,artifact_id,min_value,max_value);
        }

   fout<<"\n</Layers>"<<endl;

   system("PAUSE");
        return 0;
}

void readfile(fstream& fin,char title[],char attrib[][50], float attrib_value[][50],string
name[],int& total_row,int& total_col,float min_value[],float max_value[])
{
```

```
	char scanner;
int i=0,j=0,k=0,row=0,col=0;

fin>>ws;
fin>>title;
fin>>ws;

	///scanning for attribute text///
fin.get(scanner);
	do
{
	 j=0;
	 fin>>ws;
 while(scanner!=' '&&scanner!='\n'&&scanner!='\t')
 {
	  attrib[i][j]=scanner;
	  fin.get(scanner);
	  j++;

 }
	 i++;
	 fin.get(scanner);
	 if(scanner==' '||scanner=='\t')
	 {
	   i--;
	 }
}while(scanner!='\n'&&scanner!='\t');
total_col=i;

	///inputing for module names and module attribute value///
	do
	{
		if(fin.eof())
		{
			break;
		}
		fin.get(scanner);
		cout<<endl;
		fin.putback(scanner);

		fin>>name[k];
		cout<<name[k]<<"   ";
		for(col=0;col<total_col;col++)
		{
			fin>>ws;
			fin>>attrib_value[row][col];
			if(attrib_value[row][col]>max_value[col])
			{
				max_value[col]=attrib_value[row][col];
			}
```

```
                        if(attrib_value[row][col]<=min_value[col])
                        {
                                min_value[col]=attrib_value[row][col];
                        }
                        cout<<attrib_value[row][col]<<"   ";
                }
                k++;
                row++;
                fin>>ws;
        }while(!fin.eof());

        total_row=row;

}

void normalize(float attrib_value[][50],int total_row,int total_col,float min_value[],float
max_value[])
{
        for(int i=0;i<total_row;i++)
        {
                for(int j=0;j<total_col;j++)
                {

        attrib_value[i][j]=(attrib_value[i][j]+abs(min_value[j]))/(max_value[j]+abs(min_valu
e[j]));
                }
        }
}

void printformat(fstream& fout,char title[],char attrib[][50], float attrib_value[][50],string
name[],int total_row,int total_col,int& artifact_id,float min_value[],float max_value[])
{
        int i=0,j=0,k=0;

   fout<<"\t<Layer>"<<endl;
   fout<<"\t\t<Label>"<<title<<"</Label>"<<endl;
        for(i=0;i<total_row;i++)
        {
                fout<<"\t\t<Artifact>"<<endl;
                fout<<"\t\t\t<Label>"<<name[i]<<"</Label>"<<endl;
                fout<<"\t\t\t<UID>"<<artifact_id<<"</UID>"<<endl;
                for(j=0;j<total_col;j++)
                {

        fout<<"\t\t\t<"<<attrib[j]<<">"<<attrib_value[i][j]<<"</"<<attrib[j]<<">"<<endl;
                }
                fout<<"\t\t</Artifact>"<<endl;
        artifact_id++;
        }
        fout<<"\t</Layer>"<<endl;
        }
```

VITA

Tsz Muk (Jimmy) Kung was born in Shanghai, China, on September 27, 1980. After completing high school in Austin, Texas, he entered Texas State University-San Marcos. In 2002 he received his Bachelor's degree with a double major in Computer Science and Mathematics. Continuing his pursuit of knowledge, he completed his Master's degree in Computer Science in 2006.

Permanent Address: 3/F No. 34 Tung Lo Wan RD, Hong Kong, China.

This thesis was typed by Tsz Muk (Jimmy) Kung