ASSESSING SeeIT 3D, A SOFTWARE VISUALIZATION TOOL

by

Shalini Gadapa

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Computing and Information Systems

YOUNGSTOWN STATE UNIVERSITY

May, 2012
ASSESSING SeeIT 3D, A SOFTWARE VISUALIZATION TOOL

Shalini Gadapa

I hereby release this thesis to the public. I understand that this thesis will be made available from the OhioLINK ETD Center and the Maag Library Circulation Desk for public access. I also authorize the University or other individuals to make copies of this thesis as needed for scholarly research.

Signature:

______________________________
Shalini Gadapa, Student

Approvals:

______________________________
Bonita Sharif, Thesis Advisor

______________________________
John Sullins, Committee Member

______________________________
Yong Zhang, Committee Member

______________________________
Peter J. Kasvinsky, Dean of School of Graduate Studies and Research
Abstract

Software is inherently complex. This is especially true for large open-source systems. Over the past two decades there has been a number of software visualization tools proposed in the literature. The main idea behind creating a software visualization tool is to help a developer or maintainer comprehend the system at different levels of abstraction. Most of the tools have focused on creating elaborate and pretty looking visualizations. There have not been many cases where a tool is systematically empirically validated to make sure that it is really useful to a developer. This thesis tries to bridge this gap between the tool and its empirical validation by assessing one such software visualization tool, SeeIT 3D. Sixteen different tasks are developed in the context of understanding an open-source system, JFreeChart, written in Java. Ten subjects were recruited and an observational study was performed. The main goal was to determine the effectiveness of SeeIT 3D while performing typical software tasks when using the visualization within the Eclipse IDE. Results and observations are presented. These results will be provided as feedback to the tool developers, who may use it in further improving SeeIT 3D.
Acknowledgements

I would like to express my immense gratitude to my research advisor Dr. Bonita Sharif for her constant support and encouragement throughout my research work and preparation of the thesis. I would like to thank my committee members Dr. John Sullins and Dr. Yong Zhang for their valuable advice and encouragement.

I would like to thank my family, especially my mother and my father for their emotional support and encouragements during this period. I owe a lot to my brother and sisters for always calling to find out how life was treating me in a far away land. I would like to give special regards to my all my friends who encouraged me in this educational journey.

I truly believe that all these would have not come to pass without the financial support of the YSU School of Graduate Studies and the esteemed faculty members of YSU Computer Sciences and Information Systems (CSIS) Department.
# TABLE OF CONTENTS

LIST OF FIGURES .................................................................................................................. VIII

LIST OF TABLES ..................................................................................................................... X

CHAPTER 1  INTRODUCTION........................................................................................................ 1

1.1 Motivation ......................................................................................................................... 2

1.2 Contributions ..................................................................................................................... 2

1.3 Organization ....................................................................................................................... 3

CHAPTER 2  OVERVIEW OF SOFTWARE VISUALIZATION .................................................. 4

2.1 What is Software Visualization? ....................................................................................... 4

2.2 Software Visualization Tools .......................................................................................... 6

2.2.1 2D Tools ........................................................................................................................ 7

2.2.2 3D Tools ........................................................................................................................ 8

2.3 Empirical studies on Software Visualization Tools ....................................................... 9

CHAPTER 3  INTRODUCING SEEIT 3D .................................................................................. 12

3.1 SeeIT 3D Metaphor .......................................................................................................... 12

3.2 SeeIT 3D Internal Construction ....................................................................................... 13

3.3 Metrics Used in SeeIT 3D ............................................................................................... 14

3.4 Representing Relationships between Containers ............................................................ 16

3.5 User interactions .............................................................................................................. 17

3.6 Visualizing JFreeChart in SeeIT 3D .............................................................................. 19
CHAPTER 4  THE STUDY................................................................. 27

4.1 Experiment Design........................................................................................................ 27
4.2 Data collection................................................................................................................ 28
4.3 Participants.................................................................................................................... 28
4.4 Subject System............................................................................................................. 31
4.5 Experiment Questions................................................................................................. 31
4.6 Running the Study....................................................................................................... 33

CHAPTER 5  RESULTS AND ANALYSES .............................................. 35

5.1 Accuracy ...................................................................................................................... 35
5.2 Time ............................................................................................................................ 38
5.3 Preference Ratings ..................................................................................................... 41
5.4 Difficulty and Confidence Ratings ............................................................................. 43
5.5 Post Questionnaire Results ....................................................................................... 44
5.6 Observations ............................................................................................................... 47
5.7 Threats to Validity ...................................................................................................... 48

CHAPTER 6  CONCLUSIONS AND FUTURE WORK.............................. 49

APPENDIX STUDY MATERIAL ................................................................................. 50

A.1. Pre Questionnaire..................................................................................................... 50
A.2. Main Study Questionnaire ..................................................................................... 52
A.3. Post Questionnaire ................................................................................................. 65
A.4. Preference Ranking................................................................................................. 67
A.5. SeeIT 3D Tutorial Used in the Study ................................................................. 68

REFERENCES ........................................................................................................... 74

Institutional Review Board approval ........................................................................ 78
LIST OF FIGURES

Figure 1. A container from the DrJava system visualized in SeeIT 3D .......................... 13
Figure 2. SeeIT 3D Internal Architecture (taken from [Ramírez 2010]) ......................... 14
Figure 3. Polycylinders representing a) common base b) lines c) arcs relationships ...... 17
Figure 4. The General layout Pattern of SeeIT 3D .......................................................... 18
Figure 5. Representing JFreeChart at different granularity levels ................................. 19
Figure 6. The user customizable area ............................................................................. 20
Figure 7. Polycylinders in sorted order ......................................................................... 20
Figure 8. SeeIT 3D Preferences .................................................................................... 21
Figure 9. Visualizing JFreeChart by mapping the McCabe complexity metric to the
   Height and LOC to Color. a) The granularity level is set to method. b) The
   granularity level is set to package ............................................................................. 22
Figure 10. Lack of Cohesion mapped to the width/cross section of polycylinder which is
   a class in the org.jfree.chart package ..................................................................... 23
Figure 11. Polycylinders at the class level granularity for JFreeChart ......................... 24
Figure 12. Visual Relationship Types in a) common base b) lines and c) arcs.
   Relationships from the package org.jfree.chart.axis are being shown.................... 25
Figure 13. The JFreeChart system visualized with the McCabe Complexity representing
   color and LCOM representing height ....................................................................... 26
Figure 14. Descriptive Statistics for Participants: Experience in Java and Programming
   (Number of Years) ...................................................................................................... 29
Figure 15. Descriptive Statistics for Participants: Design and Coding Rating (Self assessment) ................................................................. 30
Figure 16. A typical setup in the study lab. ......................................................... 34
Figure 17. Accuracy box plot for Overview questions ........................................... 36
Figure 18. Accuracy box plot for Relationship questions ................................. 36
Figure 19. Accuracy box plot for Relationship+metrics questions .................... 37
Figure 20. Accuracy box plot for Metric questions ............................................. 37
Figure 21. Time box plot for the first six questions answered by all participants ... 39
Figure 22. Average time taken by participants who answered Q7 through Q16 ... 41
Figure 23. Descriptive statistics for preference ratings: 1=Strongly agree, 2= Somewhat agree, 3= Don’t agree. Refer to Appendix A.4. for each preference .......... 42
Figure 24. Descriptive statistics for Familiarity with SeeIT 3D and JFreeChart ....... 45
Figure 25. Descriptive statistics for Sufficient Time, Question Clarity, and Ease of use 46
Figure 26. Descriptive statistics for the usefulness of the tutorial and it was referred to while doing the study. ....................................................... 46
LIST OF TABLES

Table 1. The five dimensions of software visualization ....................................................... 5
Table 2. Questions and their corresponding categories ......................................................... 32
Table 3. Average time spent on Q7 through Q16 ................................................................. 40
CHAPTER 1
INTRODUCTION

Software systems are increasingly large and complex. Software is no longer written in a single closed space but is getting more and more distributed across several continents. This makes programming, analyzing, and understanding software much more difficult. Software visualization [Petre, Quincey 2006] [Zhang 2003] tools have been developed to help a developer to make the process of comprehending code easier. There have been many software visualization tools proposed in the literature over the past two decades [Bassil, Keller 2001; Teyseyre, Campo 2009]. Most of these tools do not provide a thorough empirical validation on the usefulness of the tool to the person who will be actually using the tool. In most cases, the authors of the tool paint a pretty picture of the tool along with screenshots of the tool being run on different systems. These pictures do look appealing but are they useful? This is the question we are trying to address.

There is a huge gap between the number of software visualization tools developed and the number of empirical studies conducted to determine if a software visualization tool is actually useful to the end user. This situation is slowly improving but we still have a long way to go. In this case, the end user is the developer/maintainer of the system. It is more important for a software visualization tool to be useful to a developer and in the process, if it does have aesthetically pleasing visualizations, we find it a plus. In other words, we would like to see functionality over just pure aesthetics.
1.1 Motivation

Due to the dearth of empirical validation of software visualization tools, there is a need to gather empirical evidence to show that software visualization tools actually work and help developers be more productive in their day-to-day tasks. This thesis tries to bridge the gap between a 3D visualization tool and its empirical validation. After a thorough analysis of available 2D and 3D software visualization tools, we decided to contact the authors of one such 3D software visualization tool, namely SeeIT 3D [Ramírez 2010] and asked if there was any empirical evaluation done on it. They stated that it was not done yet but they would like to get it done in the near future. This provided motivation to move forward to get the validation done.

Software visualizations can be static or dynamic. Static visualizations (similar to static analysis) do not require the program to be executing. Dynamic visualizations visualize a program in action. SeeIT 3D is a static software visualization tool.

1.2 Contributions

The approach taken in this thesis was to first study several software visualization tools (both 2D and 3D) from the literature over the past two decades. In the process, a set of potential 2D and 3D tools were gathered as candidates to conduct an empirical study. Since there are not many 3D tools [Teyseyre, Campo 2009] or their empirical validation, it was decided to focus on empirically validating a 3D software visualization tool, namely SeeIT 3D [Ramírez 2010]. We did let the authors of the tool know that we were planning on doing this.
The main contribution of this thesis is an empirical study that assesses the usefulness of one 3D software visualization tool, SeeIT 3D in the context of an open-source system namely, JFreeChart.

1.3 Organization

This thesis is organized as follows. The next chapter gives a brief introduction to software visualization including the most popular software visualization tools. Following that, Chapter 3 goes in depth into SeeIT 3D, the tool we will be assessing. Chapter 4 discusses details of our study setup and design. Chapter 5 presents observations and results. Chapter 6 concludes the thesis and presents future work.
CHAPTER 2
OVERVIEW OF SOFTWARE VISUALIZATION

This chapter presents an overview of software visualization. A selected list based on the literature search is presented along with any empirical studies that were conducted on these tools.

2.1 What is Software Visualization?

Software visualization [Price, Baecker, Small 1993] uses visual representations and visual metaphors to make software easier to comprehend. A metaphor should be effective and expressive. Software visualization derives a lot of ideas directly from information visualization [Card, Mackinlay, Shneiderman 1999]. In this case, information is the software system that consists mainly of software artifacts such as code, design, build, testing to name a few. Price et al. define software visualization as “the use of the crafts of typography, graphic design, animation, and cinematography with modern human-computer interaction technology to facilitate both the human understanding and effective use of computer software”.

When developing a software visualization tool, it is important to know what task the tool is attempting to solve. Maletic et al. [Maletic, Marcus, Collard 2002] discuss one such view of task-oriented software visualization. The tasks could be development, debugging, reverse engineering, design, and testing (this list is not exhaustive). They identify five dimensions of software visualization. They are mentioned in the table below followed by a brief explanation.
Table 1. The five dimensions of software visualization

<table>
<thead>
<tr>
<th>Five Dimensions</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks</td>
<td>Why is the visualization needed?</td>
</tr>
<tr>
<td>Audience</td>
<td>Who will use the visualization?</td>
</tr>
<tr>
<td>Target</td>
<td>What aspects of the software are to be represented?</td>
</tr>
<tr>
<td>Representation</td>
<td>How will it be represented?</td>
</tr>
<tr>
<td>Medium</td>
<td>Where will it be represented?</td>
</tr>
</tbody>
</table>

Tasks: The question itself gives an impact on software engineering in phases like debugging, programming and testing etc. And also there involves some maintenance aspects such as reverse engineering, re-engineering etc.

Audience: Most of this is defined by using attributes of the visualization system. It is generally driven by systems developers, designers, testers etc... They are the main stakeholders of the visualization.

Target: The target is concerned with the data source of the visualization. It is targeted on work products and artifacts such as designs, algorithms, source code, execution/trace information and metrics information.

Representation: It defines how one will construct the visualization. This is where the tool author needs to think of an appropriate metaphor to map the analyzed information to the available visual property.
Medium: In most cases the medium is the screen. This answer the question as to where the visualization is rendered. It could be the case that the visualization requires a certain screen size or higher resolution multiple screens joined together to be effective.

Once a tool developer answers the above five questions, they need to think in terms of features they would like the tool to have. Young et al. [Young, Munro] presents a set of properties that are desirable for a software visualization tool. Some criteria include simple navigation, low complexity, good use of metaphors and resilience to change. However, do software visualization tool authors take these criteria into account? This is the question we are partially trying to address.

Storey et al. [Storey, Fracchia, Müller 1997] and Card et al. [Card, Mackinlay, Shneiderman 1999] have discussed cognitive elements that need to be included in a tool that support a better mental model during visualizing software.

2.2 Software Visualization Tools

This section presents selected software visualization tools surveyed before conducting the actual study. A note should be made that the Unified Modeling Language (UML) [Booch, Rumbaugh, Jacobson 2005] is also a software visualization method for software. However, we were interested in looking at custom-built visualization tools that provide more semantic information from software such as metric data for classes, methods, and packages. We first present the 2D tools followed by the 3D tools surveyed. All these tools are open-source and based on academic research.
2.2.1 2D Tools

SHrimP (Simple Hierarchical Multi-Perspective) [Storey, Best, Michaud 2001] is a visualization technique to enhance how people understand complex software. It supports multiple views (graphical and textual) in the form of a nested graph. The hierarchies in programs are represented via nested graphs. A software system’s architecture can be visualized by using its package and class structure. It also shows dependencies in the code. SHrimP supports three types of zooming: geometric, semantic, and fisheye. The authors have also integrated SHrimP views with IBM Webspere Studio Application Developer Integration Edition [Rayside et al. 2003]. Even though SHrimP was designed to visualize software it has evolved to visualizing any type of structured content. There have not been any studies done to evaluate the usefulness of SHrimP.

CodeCrawler [Lanza 2003; Lanza, Ducasse 2005] is a language independent visualization tool that supports various types of software visualization views such as a complexity view, a class blueprint view, hotspot view and an evolution matrix view among others. Views can also be customized by the user on a large set of metrics. The visualization is based on a polymetric view. CodeCrawler used to be solely 2D but recently it has incorporated 3D into the views but the authors admit that they still have to assess the benefits of the additional dimension. CodeCrawler runs as a standalone application. No details on studies done on CodeCrawler are reported. See Section 2.3 for more details.

XRay [Malnati 2007] is presented as an Eclipse plug-in. It works mainly with Java code and visualizes realationships. XRay provides different views and uses different

\footnote{http://www.thechiselgroup.org/shrimp}
metrics for the views. They claim that the user can analyze the system and spot defects with their visualization. Two categories of metrics used are project based and design based. Project-based metrics are ones that are mainly important to a project manager and deal with what it takes to get to a milestone in the software cycle. On the other hand, design-metrics deal with the size and quality of a particular object such as a class. It displays different polymetric views on system complexity, class dependencies and package dependencies. It uses geometrical shapes like rectangles and lines as well as a few colors to show dependencies. The only validation done by this tool was the authors analyzing the views and saying that they make sense. Unfortunately the authors are not the end users of the tool. This type of validation consists of a significant threat to validity.

2.2.2 3D Tools

CodeCity [Wettel, Lanza 2008] is an interactive 3D visualization tool for large software systems. It is language independent. It uses a city metaphor where it shows classes as buildings and packages as districts in a software city. The tool was visualized on several large industrial systems. The same research group that developed CodeCrawler are also responsible for CodeCity. They extend the polymetric views they used earlier into 3D and map metrics on all three dimensions of the buildings. If the number of attributes of a class is mapped to the width and length of buildings and the number of methods of a class is mapped to the height, the tall buildings represent classes with more functionality. The empirical validation done was to visualize large systems like Azureus, JHotdraw, and ArgoUML. No comprehensive studies were conducted.
Codstruction\(^2\) [CS] is an interactive 3D visualization tool built as an Eclipse plugin. It has been inspired by CodeCity. It is still in the initial stages of development. Codstruction maps class hierarchy metrics.

SeeIT 3D [Ramírez; 2010] is also incorporated as an Eclipse plugin. It analyzes source code, generates certain metric information and visualizes the system in containers. A more detailed description of SeeIT 3D is given in Chapter 3 since this is the tool that was chosen to conduct the study on in this thesis. SeeIT 3D is compared to other similar tools such as XRay and Codstruction both of which are Eclipse plugins. The comparison is based on a framework by Storey et al that was extended by [Nino, Aponte, Montano, Collazos 2008]. No study involving any user was performed.

2.3 Empirical studies on Software Visualization Tools

This section presents studies conducted on software visualization tools. CodeCrawler [Lanza 2003; Lanza, Ducasse 2005] states that it has been empirically validated in industry case studies. However no detailed descriptions are given due to non-disclosure agreements. They used it mainly to reverse engineer industrial systems. It is unclear what tasks the system was tested on and if a systematic study was undertaken on specific software tasks. All this evidence is anecdotal.

Bassil et al. [Bassil, Keller 2001] are one of the first that conducted a large scale study with 100 participants to evaluate software visualization tools. They look at various functional, practical, and cognitive aspects that users look for in software visualization

\(^2\) http://codstruction.wordpress.com/about/
tools. The participants rated the usefulness of various aspects of tools that were presented to them. In general they found users were generally satisfied but there was still a big gap between desired aspects and aspects that were currently implemented. They conducted the study on more than 40 tools. They came up with a list of improvements for the tools. This work is closely related to the thesis. However, there are some subtle differences. The participants solely rated the tools based on some criteria. They did not answer any comprehension related question while using the tools. There is a big difference in these two approaches. The thesis presented here takes a more comprehension based approach where a participant is asked a question and their actual performance is gauged. A preference rating is also collected to support the comprehension part of the study in this thesis.

Lange et al. [Lange, Chaudron 2007; Lange, Wijns, Chaudron 2007] conduct an experiment to validate different views for UML class diagrams: MetaView, ContextView, MetricView, and UMLCityView. This is a true experiment that is similar to the study conducted in this thesis. They evaluate their views based on correctness and time. They answered a questionnaire about a system model and found results to be statistically significant with 100 participants. The correctness improved by 4.5% and time was reduced by 20%.

Wiss et al. [Wiss, Carr 1999] conduct a comparative study of three 3D information visualizations. They used 25 participants that performed three different tasks with three types of information visualization techniques: information landscape, cam tree, and information cube. They found one of the techniques: information landscape, to be better
in terms of time. They also found that custom navigation is important in 3D interfaces. They pose the following question at the end: *For what types of tasks is a 3D user interface best suited?* The tasks used in this study are comprehension related since the participants had to actually use the tool to answer them. However, it is important to note that this study was not specific to software visualizations. It is an information visualization technique [Card, Mackinlay, Shneiderman 1999].

In summary, this chapter gives a background view into software visualization. It also describes some of the tools analyzed before choosing the one to conduct the study on. Empirical studies conducted on software visualization tools are also presented.
CHAPTER 3
INTRODUCING SeeIT 3D

This chapter introduces SeeIT 3D and also gives some background as to how it evolved. The main features and functionality of the tool are described. For more detailed information we direct the reader to [Ramírez 2010].

SeeIT 3D\(^3\) is based on Source viewer 3D (sv3D) [Maletic, Marcus, Feng 2003]. The full form is Software visualization Eclipse Integrated Tool 3D. sv3D uses three dimensional polycylinders as a metaphor. sv3D’s 3D metaphor is based on the SeeSoft pixel metaphor [Eick, S., Sumner 1992]. It uses colors, height, and depth to show information. sv3D is a stand-alone application and is no longer maintained at the time of this writing. sv3D supports overview of the system, user interaction, zooming and panning at various speeds, filtering, and history information. It has now evolved into SeeIT 3D. SeeIT 3D is available as an Eclipse plugin. Eclipse is an open-source integrated development environment.

3.1 SeeIT 3D Metaphor

The main metaphor used here is that of a polycylinder. A polycylinder is a three dimensional bar with a polygon base. Polycylinders can be grouped together to form a container. Polycylinders have a height, width, and color. Each polycylinder can represent different types of an artifact (class, package, method, line) depending on what is being visualized. The width, height, and color can be mapped to certain complexity

\(^3\) http://code.google.com/p/seeit3d/
metrics. See Figure 1 for an example of a container. Each polycylinder is a package in the DrJava system.

![Figure 1. A container from the DrJava system visualized in SeeIT 3D](image)

### 3.2 SeeIT 3D Internal Construction

The internal architecture as presented by Ramirez [Ramírez 2010] is shown in Figure 2. SeeIT 3D uses Java3D as its graphics engine.

The **GUI and IDE Interactions** module handles all the interactions held within the Eclipse IDE. This refers to visualization commands, shortcuts and icons. The **model generator** module generates information that is used by the core in order to build the visualizations. The model generator also performs the metric calculations. The **user feedback** module allows the core to send relevant information to the user. The **visualization properties** module describes how the visualization will be rendered. This
includes color mapping, metrics mapping and the type of visual relationship (lines, arc, common base, or movement) that will be used.

![Figure 2. SeeIT 3D Internal Architecture (taken from [Ramírez 2010])](image)

### 3.3 Metrics Used in SeeIT 3D

The metrics used are lines of code (LOC), lack of cohesion (LCOM), Mc Cabe’s Complexity, and Control structure. A container can be a package or a class. Polycylinders can map to classes, methods or lines. The height of a polycylinder could
map to LOC, LCOM, or Mc Cabe’s complexity. The width could map to LCOM or Mc Cabe’s complexity. Color could map to Mc Cabe’s complexity, LOC, or control structures. Some of the combinations for the values of (container, polycylinder, height, width, color) might make less sense than others. It is important to determine what the user wants to visualize and choose the appropriate metric accordingly.

LOC refers to the number of lines of code. This could be lines of code in the package, class, or method depending on what is being represented in the polycylinder at the time the visualization is drawn.

LCOM refers to the lack of cohesion metric. It is based on a formula given in the literature and is calculated based on [Henderson-Sellers, Constantine, Graham 1996]. LCOM does not make sense at the method and package level. It only makes sense for a class to determine if the class lacks cohesiveness.

Mc Cabe’s complexity [McCabe, Butler 1989] is another metric based on a formula given in the literature. The metric gives an average value for methods and classes based on the number of linear independent paths through a program. It is based on the fact that programs that have many branches are harder to comprehend. It treats the program as a graph and measures the connections between nodes/decision places in the graph.

For more details on LCOM and Mc Cabe’s complexity refer to [Chidamber, Kemerer 1994]. Control structure indicates if there is a control structure present in code. The value can be in the following set (for, while, if, else, none). LOC, McCabe
complexity, and LCOM are all numeric values where as control structures are nominal values.

3.4 Representing Relationships between Containers

There are several ways to display relationships between containers in SeeIT 3D. They are common base, lines, arc, and movement. If common base is chosen, it shows a base under each container. The source of relationship container is shown in a darker color than the others. Lines and arc are similar. Relationships are drawn and lines or arcs from a source container to several destination containers. This representation tends to clutter the view at times, especially if there are a lot of containers related to the source. Movement is used to highlight containers that are related to the source. This view lets you view more information and still see related containers. See Figure 3 for an example of the three relation types. Movement cannot be shown in a static picture but each of the three containers shown would be slowly moving back and forth.
3.5 User interactions

Since the tool is an Eclipse plug-in, there are two types of interactions: Eclipse IDE interactions and Visualization interactions. The tool is easily accessible via the SeeIT 3D perspective in Eclipse. This is done by clicking the Open Perspective in the Eclipse menu and selecting the SeeIT 3D option. A right click on any package or class in the package explorer in Eclipse can launch the SeeIT 3D visualization by choosing the “Visualize in SeeIT 3D” from the context menu. See Figure 4 for the different areas related to the visualization.
The package explorer shows the open source system JFreeChart currently as the open project. The user customizable area is where the user can set metrics to polycylinders, change the color scale used in the visualization, determine the granularity level of the visualized containers and choose to draw relationship types in arcs, lines, or movement form as discussed in Section 3.4. Double clicking on a polycylinder opens the source code editor in Eclipse.

The visualization area is where all the containers and their relationships are drawn. The action view is a set of toolbar icons that let you select all the items in the visualization, delete the visualization, zoom, save the visualization and toggle the view between the package explorer and the visualization (a very useful feature). The SeeIT 3D tutorial has a list of what each of these commands do. Refer to the Appendix for the
tutorial used in the study. Finally the container feedback area tells you what container is currently selected and also gives the metric values for that container.

3.6 Visualizing JFreeChart in SeeIT 3D

This section will familiarize the reader with the functionality of SeeIT 3D using JFreeChart. As discussed in Section 4.4, JFreeChart is used as the subject system to test SeeIT 3D. In the figure below, JFreeChart is visualized at different granularity levels in each container. The level of granularity gets finer grained as you move from left to right; with the left most being the package and the right most being lines. In the visualization on the left, the entire JFreeChart system is visualized.

![Image of JFreeChart visualization](image)

**Figure 5. Representing JFreeChart at different granularity levels.**

The user interaction view is shown in more detail in Figure 6. The available metrics section will show what metrics are available in the current context. The user needs to drag the metrics they need and drop them into the color, height, and cross
section boxes if they want to map that metric to color, height, or cross section of the polycylinder.

![Granularity Level](image)

**Figure 6. The user customizable area**

Using the polycylinder sorting and polycylinder transparency options, one can selectively filter and detect anomalies [Ramírez 2010]. The sorting of polycylinders is shown in Figure 7.

![Figure 7. Polycylinders in sorted order](image)

The user can also set preferences in SeeIT 3D. See Figure 8 for the dialog box letting the user select the preferences. The containers per row determines how many
containers are shown on a row. Polycylinders per row determines how many cylinders are shown in a row within a container. Highlight color refers to the color of the polycylinder when it is selected. Relation mark color is the color of the arcs, lines, and base when visual relationships are drawn.

![Image of SeeIT 3D Preferences](image)

**Figure 8. SeeIT 3D Preferences**

In Figure 9 a) below, the entire JFreeChart system is visualized at the method level granularity. The same information is shown at the package level granularity (coarser grained) in Figure 9 b).
Figure 9. Visualizing JFreeChart by mapping the McCabe complexity metric to the Height and LOC to Color. a) The granularity level is set to method. b) The granularity level is set to package.
In the next figure (See Figure 10), the width of the polycylinder is mapped to the LCOM (lack of cohesion) metric. The width is also known as the cross section of the polycylinder.

![Diagram showing LCOM mapping](image)

**Figure 10. Lack of Cohesion mapped to the width/cross section of polycylinder which is a class in the org.jfree.chart package.**

In the next figure, Figure 11 each polycylinder is mapped to a class in the JFreeChart system. The LOC and McCabe complexity metrics are mapped to color and height respectively. The LCOM is mapped to the cross section (width) of the polycylinder.
Figure 12 shows all classes related to the package org.jfree.chart.axis in three different ways. Notice how the line and arc based relations are not very accurate and don’t always end at a container.
In Figure 12, McCabe complexity is mapped to color. Red indicates the most complex class and blue the least complex. Green falls somewhere in the middle. As seen from the figure, there is one class that is colored red indicating high complexity.

Figure 12. Visual Relationship Types in a) common base b) lines and c) arcs. Relationships from the package org.jfree.chart.axis are being shown.
Figure 13. The JFreeChart system visualized with the McCabe Complexity representing color and LCOM representing height.

In summary, this chapter gives an overview of SeeIT 3D showing the features in the context of JFreeChart.
CHAPTER 4

The Study

This chapter describes details about the study conducted. The experiment design, data collection, subject demographics, subject system used, experiment tasks and running of the experiment are presented.

4.1 Experiment Design

The study was setup as an observational study. We did not have a specific hypothesis as to whether or not SeeIT 3D was useful to a developer. We wanted to observe participants as they used the tool to come up with observations on tool usage.

The purpose of the study is to verify the usefulness of SeeIT 3D and to gather more information on usability in the process. Before the experiment was conducted, we applied for IRB approval and took the necessary training needed to conduct human-based research studies. There were two parts to the study: comprehension and preference. In the comprehension part, we asked the participants questions that involved understanding the tool. In this part, they had to interact with the tool to answer the questions. In the preference part, which was held after the comprehension part, they rated SeeIT 3D based on a set of eighteen criteria. The criteria chosen are a subset from what was used in [Bassil, Keller 2001]. See Section 2.3 for how the study presented in this thesis differs from Bassil et al.’s study.

For each question in the comprehension part, they timed themselves by recording the start and end time for each question. They also rated the difficulty level and
confidence in their answer for each question. The difficulty level had values of easy, average and difficult. The confidence had values of high and low.

4.2 Data collection

The study was run as a timed-response questionnaire study. The participants recorded the start and end time for each question based on the computer time. The data was collected in two forms: written and video recordings. The participants wrote the answers for each question in the comprehension part of the study on paper. They were encouraged to think-aloud. Their interaction with the tool was also captured to be analyzed later. The video recordings act as additional evidence to support what the participant was doing/interacting with when they answered the question.

4.3 Participants

There were ten volunteers from the Department of Computer Science and Information Systems that participated in the study. The students were from Computer Science, Computer Information Systems, and Electrical Engineering majors. All of the students were familiar with the process of software development. There were five females and five males. Among the participants, there were two undergraduates and eight graduate students that participated.

Each participant was asked to fill out a pre-questionnaire before they took the study. See the Appendix for the entire pre-questionnaire used. Six of the participants were very familiar with the Eclipse IDE (SeeIT 3D is used as an Eclipse plugin), the rest
were familiar with either Visual Studio or Netbeans with only one participant not using an IDE. None of the participants used a software visualization tool in the past.

The participants were asked to state the number of years they had experience with Java and in programming in general. The number of years were mapped to 1, 2, or 3 based on the following scale. Experience between 1 to 3 years was mapped as 1; between 3 to 5 years was mapped as 2, between 5-10 years was mapped as 3. See Figure 14 for boxplots on Java and programming experience. Most of the participants had between 1 and 5 years of experience in using Java as well as programming in general. There was more variation in programming experience than there was in Java experience. This was probably due to the fact that a lot of the courses that students take use Java as the preferred programming language.

![Boxplot of Java and Programming Experience](image)

**Figure 14.** Descriptive Statistics for Participants: Experience in Java and Programming (Number of Years)
The participants were asked to self-rate their design skills and coding skills. A below average rating was mapped to 1, Average was mapped to 2, Above average was mapped to 3, and Excellent was mapped to 4. See Figure 15 for boxplots on their self-rating of design and coding skills. Most of the participants rated themselves as having design skills below average while their coding skills were rated between average and above average. The variation between design and coding skills was similar as seen by the distribution in the box plot.

![Boxplots for Design and Coding Ratings](image)

**Figure 15. Descriptive Statistics for Participants: Design and Coding Rating (Self assessment)**
4.4 Subject System

SeeIT 3D works with Java systems so we needed to find a system written in Java. Instead of using a toy application, we decided to run the study on a real open-source system, namely JFreeChart [JF]. The study was conducted on JFreeChart version 1.0.14.

JFreeChart is a free Java chart library. JFreeChart supports pie charts (2D and 3D), bar charts (horizontal and vertical, regular and stacked), line charts, scatter plots, time series charts, high-low-open-close charts, candlestick plots, Gantt charts, combined plots, thermometers, dials and more. JFreeChart can be used in applications, applets, servlets and JSP. JFreeChart has 504 classes and 37 packages, 7551 methods, and 73615 lines of code within the 91174 methods of code in total. Screen shots of how JFreeChart is used with the SeeIT 3D visualization are shown in Chapter 3.

4.5 Experiment Questions

This section discusses the category of questions used in the comprehension part of the study. There were sixteen questions in this main part of the study. They were categorized into four basic types. The overview questions simply dealt with how the user goes about getting a general overview of the system. Tasks involved visualizing the entire system in a container, zooming capabilities, moving capabilities as well as being able to set preferences in SeeIT 3D. Relationship category questions consisted of showing which classes or packages were related to each other. Relationship + metrics category questions are similar to relationship category questions but with the added information of metrics requested for a class or method. Finally, the Metrics category questions were based on giving certain metric values [Chidamber, Kemerer 1994] such as
McCabe’s Complexity, LOC (Lines of code), LCOM (lack of cohesion), and control structures (if/else, while).

The reader is referred to the Appendix (A.2) for all the questions used in the study. In the table below, we present the 16 questions in brief form and also specify their category. The second column in the table simply gives the main idea of the question. The appendix gives the actual wording of the questions.

**Table 2. Questions and their corresponding categories**

<table>
<thead>
<tr>
<th>Question Number</th>
<th>Main Idea of Question</th>
<th>Question Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>simple visualization of whole system</td>
<td>Overview</td>
</tr>
<tr>
<td>2</td>
<td>change granularity to class, set height, color and cross section</td>
<td>Overview</td>
</tr>
<tr>
<td>3</td>
<td>zoom in / zoom out</td>
<td>Overview</td>
</tr>
<tr>
<td>4</td>
<td>rotate</td>
<td>Overview</td>
</tr>
<tr>
<td>5</td>
<td>expand class to method level granularity</td>
<td>Overview</td>
</tr>
<tr>
<td>6</td>
<td>expand method to line level granularity</td>
<td>Overview</td>
</tr>
<tr>
<td>7</td>
<td>set preferences - container per row, poly cylinder</td>
<td>Overview</td>
</tr>
<tr>
<td>8</td>
<td>set preferences - background and highlight</td>
<td>Overview</td>
</tr>
<tr>
<td>9</td>
<td>visualize as file relationship visual type for 1 class in a package</td>
<td>Relationship</td>
</tr>
<tr>
<td>10</td>
<td>visualize three packages state packages related to each package.</td>
<td>Relationship</td>
</tr>
<tr>
<td>11</td>
<td>name biggest related package, name most complex class with respect to Mccabe complexity</td>
<td>Relationship + Metrics</td>
</tr>
<tr>
<td>12</td>
<td>name class with largest LOC in a package</td>
<td>Metrics</td>
</tr>
<tr>
<td>13</td>
<td>for biggest class, give metric values LOC, Mccabe, Lack of Cohesion</td>
<td>Metrics</td>
</tr>
<tr>
<td>14</td>
<td>visualize package as method level, name method and class that has highest mccabe complexity</td>
<td>Relationship + Metrics</td>
</tr>
<tr>
<td>15</td>
<td>name largest method wrt Mccabe complexity</td>
<td>Metrics</td>
</tr>
<tr>
<td>16</td>
<td>visualize parent package and visualize as file. note differences in granularity in polycylinder</td>
<td>Metrics</td>
</tr>
</tbody>
</table>
4.6 Running the Study

Participants were recruited via email and word-of-mouth to participate in the study. They were told that we were conducting an experiment on understanding how people understand software visualization tools. There was no compensation provided. They were sent a short 4-page tutorial via email to go through before the study. This was not required since the tutorial was also available on the day of the study.

The overall structure of the study consists of four stages: informed consent, pre-questionnaires, experiment, and post-questionnaires. The study was conducted in a controlled lab setting at Youngstown State University. When the participants arrived they went through a short tutorial with the experimenter on a test system namely, DrJava [DrJava]. They were then asked to read an informed consent form and sign that they will be participating in the study and that they will be audio and video taped. The face of the participant was not recorded, only the screen was videorecorded along with their think-aloud comments. See the figure below for example of a typical study setup in the lab. The screen size used was 19 inches. The system used had 4GB of memory and was run on a 64-bit machine with Windows 7 installed.
After the informed consent step, they were asked to fill out a pre-questionnaire that gathered demographic information as presented in Section 4.3 above. They were asked about their computing background, familiarity with an IDE, number of years of programming experience and if they used a software visualization tool before. They were also asked to rate their design and coding skills. Refer to the Appendix (A.1) for the pre-questionnaire used in this study. Following the pre-questionnaire, they started working on the actual study. When they were done with all the sixteen questions, they were asking to do the preference rating and a short post-questionnaire (See Appendix A.3 and A.4).
CHAPTER 5

Results and Analyses

This chapter presents the results and analyses of the data that was collected from the study. The accuracy, time taken, preference ratings, difficulty and confidence ratings are given. Data from the post questionnaire is also presented. Finally, general observations are presented including desired features for SeeIT 3D.

As mentioned before we gathered 10 participants who volunteered to take the study. Also, since we had only 10 participants, we did not perform a statistical analysis on the data. We report our qualitative analysis on the findings.

5.1 Accuracy

For each question we gave the participant a score. For the first six questions, the experimenter observed the participant without interacting with them to see if they were doing the question correctly. This was needed since it involved a lot of interacting with the visualization than writing something down as the answer. A score of 0 indicates that they did not answer the question correctly. A score of 1 indicates that they answered the question correctly. A score of 0.5 indicates a partially answered question. If a participant did not answer a question, we gave them a score of 0.

Figure ranges: Figure 17 through Figure 20 give the descriptive statistics for accuracy scores for each question category (Overview, relationship, relationship+metric, and metrics).
Figure 17. Accuracy box plot for Overview questions

Figure 18. Accuracy box plot for Relationship questions
Figure 19. Accuracy box plot for Relationship+metrics questions

Figure 20. Accuracy box plot for Metric questions
As can be seen from the above box plots, the overview questions perform relatively good with question 4 done correctly by all. This question was on rotating the visualization. However, as the questions got more difficult a lot of the participants were not able to answer the questions. This was especially true for questions that asked for metrics (Q12, Q13, Q15, Q16, Q11, Q14). The questions on the relationship category showed that the lower number of packages, a container was related to, the better the answer was. This indicates that most subjects were unable to find the names of all related packages. This probably means that some mechanism needs to be in place to let the user know of names of related packages.

5.2 Time

The participant recorded the start and end time for each question. This is how we timed each question. The study took an hour on average. Since there were many participants (around 6) that did not answer the harder questions, it was not possible to note on time data for them, even though we scored them as having a zero score. We only report time for questions that were answered correctly. The first six questions were answered by all and so it makes sense to show the boxplot below. We see that question 4 on rotating was the easiest for all to do and was done in the shortest time. The hardest one in terms of time was question 2 that involved changing the granularity level and mapping metrics to polycylinders. This could be due to the fact that this question was the first one they encountered and had a small learning curve in actually learning to use the drag and drop feature. Most of them finished these questions in under two minutes.
For the rest of the questions, since a lot of participants did not answer them, we report on average time spent on them only by the people who answered the questions. We also report on how many people actually attempted these questions (See Table 3). Besides this we also report on the average score for the same participants to see if we have any accuracy/speed tradeoff. From the data, it doesn’t seem to have any. We see that question Q10ab took nearly six minutes on average to complete and most of the participants got it correct (0.833 is close to 1).

Figure 21. Time box plot for the first six questions answered by all participants
<table>
<thead>
<tr>
<th>Question</th>
<th>Number of People Attempted</th>
<th>Average Time in seconds</th>
<th>Average Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q7</td>
<td>8</td>
<td>525</td>
<td>0.625</td>
</tr>
<tr>
<td>Q8</td>
<td>5</td>
<td>336</td>
<td>1</td>
</tr>
<tr>
<td>Q9</td>
<td>4</td>
<td>345</td>
<td>0.875</td>
</tr>
<tr>
<td>Q10ab</td>
<td>3</td>
<td>580</td>
<td>0.833</td>
</tr>
<tr>
<td>Q10cd</td>
<td>4</td>
<td>225</td>
<td>1</td>
</tr>
<tr>
<td>Q10ef</td>
<td>3</td>
<td>120</td>
<td>1</td>
</tr>
<tr>
<td>Q10g</td>
<td>1</td>
<td>420</td>
<td>1</td>
</tr>
<tr>
<td>Q11</td>
<td>3</td>
<td>400</td>
<td>1</td>
</tr>
<tr>
<td>Q12</td>
<td>4</td>
<td>120</td>
<td>1</td>
</tr>
<tr>
<td>Q13</td>
<td>3</td>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>Q14</td>
<td>6</td>
<td>370</td>
<td>1</td>
</tr>
<tr>
<td>Q15</td>
<td>6</td>
<td>220</td>
<td>1</td>
</tr>
<tr>
<td>Q16</td>
<td>5</td>
<td>264</td>
<td>1</td>
</tr>
</tbody>
</table>
We now report on the preference ratings given by the participants. We only used eight participants’ data since two did not complete this part or the post questionnaire part. Most of the participants agreed that SeeIT 3D had some automatic layout capabilities (P3) as well as navigating from source code view to visualization view (P2). The highest discrepancy in rating was in P11, possibility on hiding the information on demand. This was not really a feature in SeeIT 3D. It also did not have a good rating on animation effects (P17). P10 involved saving the views for future use. This feature is available in SeeIT 3D but they were not required to exercise it during the study which might have caused them to rate it incorrectly.

Figure 22. Average time taken by participants who answered Q7 through Q16

5.3 Preference Ratings

We now report on the preference ratings given by the participants. We only used eight participants’ data since two did not complete this part or the post questionnaire part. Most of the participants agreed that SeeIT 3D had some automatic layout capabilities (P3) as well as navigating from source code view to visualization view (P2). The highest discrepancy in rating was in P11, possibility on hiding the information on demand. This was not really a feature in SeeIT 3D. It also did not have a good rating on animation effects (P17). P10 involved saving the views for future use. This feature is available in SeeIT 3D but they were not required to exercise it during the study which might have caused them to rate it incorrectly.
Figure 23. Descriptive statistics for preference ratings: 1=Strongly agree, 2=Somewhat agree, 3= Don’t agree. Refer to Appendix A.4. for each preference.
5.4 Difficulty and Confidence Ratings

After each question, the subjects were asked to rate the difficulty level and also give their confidence in their answer. Difficulty level values were Easy, Average, and Difficult. Confidence values were high or low. We only included data on subjects that answered the questions. Those that did not answer were excluded from this analysis. We will discuss difficulty levels first following by confidence.

For difficulty, we found that on average most questions were rated as Easy with the exception of two questions (Q7 and Q14). They were rated within the range of Average to Difficult. Question 7 involved setting preferences for SeeIT 3D and noticing the differences in the visualization after changing the preferences. They found the preference window difficult to understand causing them to rate the question as average. Question 14 involved naming a method and corresponding class that has the highest McCabe complexity. This question was more involved and required them to understand the toggle feature in SeeIT 3D. This feature might need to be made more explicit.

For confidence, questions Q7, Q8, Q9, and Q10ab were rated lower on average than the other questions. As mentioned earlier, a lot of students did not attempt questions 7 through 10 due to the fact that they were more involved. Question 8 involved changing background and highlight color. Even after resetting the view, the background color did not change, only the highlight color changed. This was confusing to the participants and might be the reason for rating this with low confidence. Question 9 involved naming classes and corresponding packages that were related to a particular class. Once again they needed to know that a package name is always shown as the first line in the java
source file in the source editor. Some participants did not realize this to be the case, especially the ones with low experience in programming and design. Question 10ab was also similar asking to name packages related to a particular package. The low confidence could also be due to the fact that this was the first question of this type, since consequent similar questions were rated higher.

5.5 Post Questionnaire Results

This section reports on the findings from the post questionnaire. We use the data from eight subjects for this purpose since two participants did not answer. See Appendix (A.3.) for the questions asked. With respect to the response time (Q.9) of the visualization everyone agreed that the visualization was extremely slow. With respect to whether they would use it in the future (Q.10), three people said no but a majority (57%) said they would consider using it. Q.11. asked about the overall impression of the tool. 50% said it needs significant improvement. Only one said it was a great tool.

Next, descriptive statistics and explanation is provided for the first eight questions in the post questionnaire. For SeeIT 3D familiarity, the value 1 is mapped to familiarity with the tool, 2 is mapped with unfamiliarity with the tool, 3 is mapped to used but not familiar with the tool, and 4 is mapped to the fact that they never heard of the tool. For JFreeChart familiarity, the value 1 is mapped to “very familiar with the design structure”, 2 is mapped to “heard but not familiar with the design structure”, 3 is mapped to “used it but not familiar with the design structure” and finally 4 is mapped to “never heard of JFreechart”.
See Figure 24 for the distribution. As we can see no one was familiar with SeeIT 3D. Only one subject was familiar with the design of JFreeChart. The median for both distributions is a 4 which maps to “never heard of it”. This was what we wanted because we wanted to test SeeIT 3D on people who did not use it before to see how quickly they adapt to it.

![Descriptive statistics for Familiarity with SeeIT 3D and JFreeChart](image)

**Figure 24. Descriptive statistics for Familiarity with SeeIT 3D and JFreeChart**

Participants also rated if they had sufficient time, if the questions were clear, if SeeIT 3D was easy to use, and if the tutorial helped them. These were also rated on a Likert scale from 1 = Strongly agree, 2= Somewhat agree, 3= No opinion, 4=Somewhat disagree, 5= Strongly disagree. See Figure 25 for the distribution. All agreed that they had sufficient time for the study. They found the questions clear and the tool easy to use in general. Only one person thought the tool was terrible.
Figure 25. Descriptive statistics for Sufficient Time, Question Clarity, and Ease of use

Figure 26. Descriptive statistics for the usefulness of the tutorial and it was referred to while doing the study.
We also asked participants if they thought the tutorial helped and if they referred to the tutorial during the study (See Figure 26). Most participants agreed that the tutorial helped them understand SeeIT 3D ahead of time. 50% of the people referred to the tutorial during the study.

5.6 Observations

It was observed that almost all of the participants said that the visualization was extremely slow. Half of the participants thought that the tool needed to be improved. With the drawing of the relationships visually using arcs or lines, it was immediately noticed that the line or arc drawing was incorrect. There were arcs that didn’t point to anything. A few participants who had a higher skill level (in terms of design and coding) said that they could see how this visualization would be helpful for large projects. A comment was made on having a bigger progress bar to see that the visualization was working in the background. Some participants noted that when a container was expanded to line level granularity the response time in SeeIT 3D dramatically slowed down.

Some participants did not expect previous containers to remain in the view that they created earlier and closed. They were expecting to start with a fresh empty visualization. This was a little confusing at times.

The good news is that almost 57% of them said that they would use it in the future.
5.7 Threats to Validity

We tried to minimize threats to validity wherever possible. The research participants did not know about the main goal of the research. They only knew that they would participate in helping us understand how software visualization tools work. During the study, there was minimal contact between the experimenter and the participants. The experimenter did not interact or direct the participants to complete the questions in one way or another. One possible threat to validity could be the subject pool which consists mainly of students. We also did not have many expert subjects in the pool. We are unable to generalize anything based on this study in terms of external validity. This would be possible if we had a larger subject pool and more expert subjects. It is worth noting that the expert subjects (ones who had a good background in design and coding) did not find the tool friendly to use.
CHAPTER 6
Conclusions and Future Work

The thesis presents an empirical validation of a 3D software visualization tool namely SeeIT 3D. Ten participants volunteered to take the study in a controlled lab setting. Before the choosing of SeeIT 3D, a literature search was conducted on both 2D and 3D tools to determine the best one to choose for the study. The functionality of SeeIT 3D is briefly described in the context of JFreeChart, an open-source Java chart library. Results indicate that most participants would use the tool again in the future but would like the response time to be faster. They also saw value in how the tool helps in understanding large systems. Some bugs with drawing of the visual relationship type in the form of arcs and lines were asked to be improved and were easily noticed.

For future work, the plan is to conduct an eye-tracking study analyzing more software tasks such as debugging. This will determine the kinds of tasks SeeIT 3D is better at addressing. Another direction is to compare SeeIT 3D to CodeCity to determine the advantages of each. We would also like to re-run this study with a larger number of participants with more varied programming experience. Extending the study to involve other software tasks such as impact analysis, bug fixing, new feature addition, and refactoring is also a future direction.
APPENDIX

Study Material

A.1. Pre Questionnaire

1. Are you a graduate or undergraduate student?  Graduate  Undergraduate

2. Which computing environment or platform do you use on a regular basis? Check all that apply.
   a. Linux
   b. Mac
   c. Windows
   d. Other (specify):

3. Which of the following Integrated Development Environments (IDE’s) do you use or have used in the past? Check all that apply.
   a. I do not use an IDE
   b. Visual Studio
   c. Eclipse
   d. Develop
   e. Sharp Develop
   f. Net Beans
   g. Code: Blocks
   h. Other (please specify)

4. How would you rate your software analysis and design skills?
   a. Poor
   b. Below Average
   c. Average
   d. Above Average (Good)
   e. Excellent

5. How would you rate your software coding skills?
   a. Poor
   b. Below Average
   c. Average
   d. Above Average (Good)
   e. Excellent

6. How many years of experience do you have using Java?
   a. None
   b. Between 1 and 3
   c. Between 3 and 5
   d. Between 5 and 10
   e. More than 10
7. How many years of experience do you have actively programming in any language?
   a. None
   b. Between 1 and 3
   c. Between 3 and 5
   d. Between 5 and 10
   e. More than 10

8. If you have used a particular software visualization tool, please list it.
A.2. Main Study Questionnaire

Instructions for study

- The purpose of this study is to understand how people use software visualization tools in particular SeeIT 3D. It is not to measure your skills at software visualization.
- The questions will require you to think aloud. So state what you are thinking out loud as you are going through the process of answering the questions. You will also be required to write down answers to some of the questions in the space provided.
- Please answer the questions from the perspective of a person trying to understand the system.
- You will need to record the start time before you start each question. Use computer time.
- You will need to record the end time as soon as you finish answering each question.
- For each question, you will be asked to rate your answer for it’s difficulty level and confidence in the answer you provided
- Before you start the study you will go through a short tutorial on SeeIT 3D.
- The actual study consists of 16 questions.
- The total time required for this study is 60 minutes.

The system that will be visualized in SeeIT 3D will be JFreeChart

JFreeChart

JFreeChart is a free Java chart library. JFreeChart supports pie charts (2D and 3D), bar charts (horizontal and vertical, regular and stacked), line charts, scatter plots, time series charts, high-low-open-close charts, candlestick plots, Gantt charts, combined plots, thermometers, dials and more. JFreeChart can be used in applications, applets, servlets and JSP. Some of the charts supported are shown below.
Now you will begin with the questions in the study. Remember to think out loud your choices as you go along.

During the study you will be allowed to refer to the tutorial you just went through.

Start Time: ____________

Q1) Go to Package Explorer and visualize the JFreeChart software system in SeeIT 3D. This is done via a right click on the software system. You will see the progress indicator at the bottom of the page showing SeeIT 3D working in the background.

Difficulty Level: Easy Average Difficult

Confidence: High Low

Comments (if any):

End Time: ____________

Start Time: ____________

Q2) Click on any poly-cylinder in the visualization you see. Next change the granularity level to “Class”. Set the height of the poly-cylinder to map to the “Lack of Cohesion”
metric. Set the Color to “McCabe Complexity” metric and set the cross section to “LOC” metric.

**Difficulty Level:** Easy Average Difficult
**Confidence:** High Low

What does the visualization show you now?
**Comments (if any):**

End Time: ________________

---

**Start Time:** ________________

**Q3** Try to zoom out the entire visualization that you see in front of you so that you can see the entire visualization on your screen. Next try to zoom in to a specific poly-cylinder.

**Difficulty Level:** Easy Average Difficult
**Confidence:** High Low

**Comments (if any):**

End Time: ________________

---

**Start Time:** ________________

**Q4** Try to rotate the entire visualization. Next try to move the entire visualization you see in front of you.

**Difficulty Level:** Easy Average Difficult
**Confidence:** High Low

**Comments (if any):**

End Time: ________________

---

**Start Time:** ________________

54
Q5) Pick a ‘class’ in the visualization you see. You need to expand this class into its own separate container that is at the level of method granularity. (Hint: Use the + sign from the top right toolbar). Highlight the new container you just created. Change the color scale to Rainbow.

Did the color change behave the way you expected it to?

Difficulty Level: Easy Average Difficult
Confidence: High Low
Comments (if any):
End Time: ________________________________

Start Time: ________________________________

Q6) Click on a ‘method’ in your newly created container and expand this into a line-level granularity.

How many containers do you see on the screen?

Difficulty Level: Easy Average Difficult
Confidence: High Low
Comments (if any):
End Time: ________________________________

Start Time: ________________________________

Q7)
- Delete all the containers you see in the visualization in front of you.
- Set the preferences for SeeIT 3D to customize the ‘Container per row’ to 3 and ‘Poly cylinders per row’ to 20. (Window->Preferences->See IT 3D)
Select two packages: org.jfree.chart.annotations and org.jfree.chart.axis from the Eclipse Package. (Hold Ctrl down to choose more than 1 package)

Visualize these two packages in SeeIT 3D. You will see the progress indicator at the bottom of the page showing SeeIT 3D working in the background.

Set the preferences for SeeIT 3D to customize the ‘Container per row’ to 5 and ‘Polycylinders per row’ to 10.

Reset the view to see the changes. (refer to tutorial on how to reset)

What differences did you notice in the visualization? Please write your answer below but you may also add more details out loud.

Difficult Level: Easy Average Difficult
Confidence: High Low
Comments (if any):
End Time: ________________
Start Time: ________________

Q8)

Close the 3D view in front of you.

Change the background color and the highlight color in the SeeIT 3D preferences (Window->Preferences->See IT 3D)

Visualize the class ArrowNeedle in the package org.jfree.chart.needle

Note that you will need to reset the view after visualizing (see tutorial if needed on how to reset view)

Did you expect to see three containers in the view?

What differences in the visualization did you find? Did the color changes work? Please write your answer below but you may also add more details out loud.
Start Time: __________________________

Q9)

- Delete all the containers in the visualization you see in front of you.
- Select the class `DefaultCategoryDataset` from the `org.jfree.data.category` package.
- Open the source code for the class `DefaultCategoryDataset` that you just selected.
- Visualize this class in SeeIT 3D as a File. (Hint: right click)
- In the Visualization, change the Relationship Visual Type to Common Base.
- Also, try changing the Relationship Visual Type for the class `DefaultCategoryDataset` to Lines, Arcs and Movement to see the differences.

**Name the classes and their corresponding packages that `DefaultCategoryDataset` is related to? Write your answer below.**

<table>
<thead>
<tr>
<th>Class name</th>
<th>Package name</th>
</tr>
</thead>
</table>

**Difficulty Level:** Easy    Average    Difficult
**Confidence:** High    Low

**Comments (if any):**

End Time: __________________________
Clear all the visualization containers from the screen.

Select the following packages from the Package Explorer.
- org.jfree.chart.plot
- org.jfree.chart.plot.dial
- org.jfree.chart.renderer

Visualize the above three packages in SeeIT 3D.

To answer the next sub-questions a) through g), you will need to modify the visualization to visually include the relationships to the packages you just chose.
- You can do this by choosing the particular package and then selecting one of three visualization types (Arcs, Movement, and Lines) for that package.
- At any time you can turn off the relationship visual types by choose the option “No Visual Relation” for the particular package you are interested in.

a) Name the packages that org.jfree.chart.plot.dial is related to? Write your answer below.

b) Which relationship visualization type did you use to give your answer above?

You may circle more than one if needed.

Arcs     Movement     Lines

Briefly explain why or which one you preferred. In addition you can give more out loud explanations.

Difficulty Level:   Easy    Average    Difficult
Confidence:        High    Low

Comments (if any):

End Time:__________________________
Start Time: ______________________

c) Name the packages that `org.jfree.chart.plot` is related to? Write your answer below.

d) Which relationship visualization type did you use to give your answer above?
You may circle more than one if needed.

<table>
<thead>
<tr>
<th>Ares</th>
<th>Movement</th>
<th>Lines</th>
</tr>
</thead>
</table>

Briefly explain why or which one you preferred. In addition you can give more out loud explanations.

Difficulty Level: Easy  Average  Difficult
Confidence: High  Low
Comments (if any):

End Time: ______________________
Start Time: ______________________

e) Name the packages that org.jfree.chart.renderer is related to? Write your answer below.

f) Which relationship visualization type did you use to give your answer above? 
You may circle more than one if needed.

<table>
<thead>
<tr>
<th>Arrows</th>
<th>Movement</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Briefly explain why or which one you preferred. In addition you can give more out loud explanations.

Difficulty Level: Easy Average Difficult
Confidence: High Low
Comments (if any):

End Time: ______________________

Start Time: ______________________

g) Which of the relationship visual types were more helpful to answer the above three questions? You may circle more than one.

<table>
<thead>
<tr>
<th>Arrows</th>
<th>Movement</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Difficulty Level: Easy Average Difficult
Confidence: High Low
Comments (if any):

End time: ______________________
Q11)
   a) Name the biggest package that org.jfree.chart.plot.dial is related to? State how you determined your answer.

   b) In the biggest package you identified above, what is the most complex class with respect to McCabe complexity? State how you determined your answer. (Hint: need to use the toggle link)

   Difficulty Level: Easy    Average    Difficult
   Confidence:    High     Low
   Comments (if any):

Q12) Clear all the visualizations on the screen. In the org.jfree.chart.event package, name the class that is the largest in terms of LOC (lines of code)?

   Difficulty Level: Easy    Average    Difficult
   Confidence:    High     Low
   Comments (if any):

Q13) For the biggest class you identified above, give the metric values for

   LOC: _______________________
   McCabe Complexity: _________________
Lack of Cohesion: ______________________

**Difficulty Level:** Easy Average Difficult

**Confidence:** High Low

**Comments (if any):**

**End Time:** ______________________

**Start Time:** ______________________

Q14) Visualize the `org.jfree.chart.event` package at the method level granularity. The height of the poly-cylinder should be mapped to McCabe Complexity metric.

*After you do this, can you name the method and the corresponding class that has the highest McCabe complexity metric value? (Hint: You will need to use the toggle view option in SeeIT 3D (see tutorial if needed)*

**Method name:** ______________________

*The above method is in class : ______________________

One line description of what the method does: * (Hint: You will find this in comments in the source code right above the method name)

**Difficulty Level:** Easy Average Difficult

**Confidence:** High Low

**Comments (if any):**

**End Time:** ______________________

**Start Time:** ______________________

Q15)

- Delete all the containers you see in your visualization. Be careful not to delete the source code.
• Choose the class **ChartUtilities** from the package **org.jfree.chart** and visualize it in seeIT 3D
• Pick the largest method in terms of LOC in **ChartUtilities** and give the following metrics
  
  **Name of largest method:** ______________________
  **McCabe Complexity:** ______________________
  **LOC:** ______________________
  **Difficulty Level:** Easy Average Difficult
  **Confidence:** High Low
  **Comments (if any):**
  **End Time:** ______________________

**Start Time:** ______________________

**Q16)**

• Delete all the containers you see in your visualization.
• Choose the class **AbstractCategoryItemRenderer** from the package **org.jfree.chart.renderer.category**
• Open the source file by double clicking on it.
• Next Visualize it as a File.
• Go back to the source file and now Visualize the parent package of the file **AbstractCategoryItemRenderer** (Remember to see the progress bar at the bottom right of the screen showing SeeIT 3D working in the background)
• At this point you should have two containers showing on the seeIT 3D visualization.

  a) **At what level of granularity is the class AbstractCategoryItemRenderer visualized?**

  b) **At what level of granularity is the package org.jfree.chart.renderer.category visualized?**
c) What are the possible metrics to use to visualize the class AbstractCategoryItemRenderer?

d) What are the possible metrics to use to visualize package org.jfree.chart.renderer.category?

<table>
<thead>
<tr>
<th>Difficulty Level:</th>
<th>Easy</th>
<th>Average</th>
<th>Difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidence:</td>
<td>High</td>
<td>Low</td>
<td></td>
</tr>
</tbody>
</table>

Comments (if any):

End Time: ____________________________
A.3. Post Questionnaire

1. Were you familiar with SeeIT 3D?
   a. I am very familiar with the tool
   b. I heard of it but I am not familiar with the tool
   c. I used it but I am not familiar with the tool
   d. I have never heard of it.

2. Were you familiar with JFreeChart?
   a. I am very familiar with the design structure
   b. I heard of it but I am not familiar with the design
   c. I used it but I am not familiar with the design
   d. I have never heard of it.

3. I had sufficient time to complete the questions.
   a. Strongly agree
   b. Somewhat agree
   c. Not certain / No opinion
   d. Somewhat disagree
   e. Strongly Disagree

4. The questions were clear to me.
   a. Strongly agree
   b. Somewhat agree
   c. Not certain / No opinion
   d. Somewhat disagree
   e. Strongly Disagree

5. I was able to easily visualize information in SeeIT 3D
   a. Strongly agree
   b. Somewhat agree
   c. Not certain / No opinion
   d. Somewhat disagree
   e. Strongly Disagree

6. SeeIT 3D was easy to use
   a. Strongly agree
   b. Somewhat agree
   c. Not certain / No opinion
   d. Somewhat disagree
   e. Strongly Disagree

7. The tutorial provided before the study helped me understand the concepts in the study
   a. Strongly agree
   b. Somewhat agree
   c. Not certain / No opinion
   d. Somewhat disagree
c. Strongly Disagree

8. The following describes how often I referred to the tutorial while doing the study
   a. Frequently
   b. Occasionally
   c. Rarely
   d. Never

9. Overall, what was the response time of the visualization? How long did you have to wait?
   a. Fast response (less wait time)
   b. Slow response (more wait time)

10. In the future, would you consider using SeeIT 3D if you needed to visualize a large software system in Java?
    a. Yes
    b. No

11. What was your overall impression of the tool?
    a. Great tool
    b. Needs improvement
    c. Hated it

12. What are your suggestions to improve the SeeIT 3D tool?

13. Please list any additional comments you have:
### A.4. Preference Ranking

Please rate the usefulness of the functional aspects and features of SeeIT 3D in the column below.

<table>
<thead>
<tr>
<th>Strongly agree = 1</th>
<th>Somewhat = 2</th>
<th>Don’t agree = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ID</strong></td>
<td><strong>Tool Functional Aspects</strong></td>
<td><strong>SeeIT 3D Conformance to the aspects</strong></td>
</tr>
<tr>
<td>P1</td>
<td>Source code Visualization (textual views)</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>Navigating from source code view to the visualization view</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>Automatic layout capabilities</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>Automatic scaling capabilities</td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td>Abstraction mechanisms (e.g., display of subsystem nodes)</td>
<td></td>
</tr>
<tr>
<td>P6</td>
<td>Multiple views where same object is highlighted in all views.</td>
<td></td>
</tr>
<tr>
<td>P7</td>
<td>Zoom In and Zoom Out capabilities</td>
<td></td>
</tr>
<tr>
<td>P8</td>
<td>Use of Colors</td>
<td></td>
</tr>
<tr>
<td>P9</td>
<td>Navigation across Hierarchies (Subsystems…)</td>
<td></td>
</tr>
<tr>
<td>P10</td>
<td>Saving the views for future use</td>
<td></td>
</tr>
<tr>
<td>P11</td>
<td>Possibility on hiding the information on demand</td>
<td></td>
</tr>
<tr>
<td>P12</td>
<td>Possibility to customize visualization</td>
<td></td>
</tr>
<tr>
<td>P13</td>
<td>Hierarchical representing of classes and sub classes</td>
<td></td>
</tr>
<tr>
<td>P14</td>
<td>Synchronization Browsing Views of Graphic and textual (Same type or different type)</td>
<td></td>
</tr>
<tr>
<td>P15</td>
<td>Graph Visualization</td>
<td></td>
</tr>
<tr>
<td>P16</td>
<td>Overviewing of the hierarchical structure and visualized software</td>
<td></td>
</tr>
<tr>
<td>P17</td>
<td>Animation effects</td>
<td></td>
</tr>
<tr>
<td>P18</td>
<td>3D representation of layouts</td>
<td></td>
</tr>
</tbody>
</table>
A.5. SeeIT 3D Tutorial Used in the Study

A major part of this tutorial was taken from the SeeIT 3D userguide\(^4\).

This tutorial will describe SeeIT 3D a software visualization tool. It will be demonstrated by visualizing the DrJava open source software system.

**DrJava**

DrJava is a lightweight development environment for writing Java programs. It is designed primarily for students, providing an intuitive interface and the ability to interactively evaluate Java code. It also includes powerful features for more advanced users. DrJava is available for free under the BSD License, and it is under active development by the JavaPLT group at Rice University. (http://www.drjava.org/)

DrJava currently has 29 packages, 782 classes, 7137 methods, 63683 lines of code within methods and 89021 lines of code in total.

**SeeIT 3D Introduction**

SeeIT 3D is a software visualization tool for the Eclipse IDE. It allows Java developers to visualize and analyze information about a software project.

A basic set of operations can be applied to the graphical elements displayed in the SeeIT 3D view. So the graphic scene can be explored and manipulated by the user, for example, by rotating, translating and zooming containers, as well as, the whole world where the elements are exposed.

- To translate (move) the world, simply right click on a place where no containers are present and drag the mouse in the desired direction. To rotate the world, use left button of the mouse.
- These two operations can be also performed on each of the containers in the view. To do so, select a container (or multiple holding the Ctrl key) and translate it using the right button or rotate it using the left button.
- For zooming in or out: the wheel of the mouse needs to be used.

SeeIT 3D overview in an IDE

**Steps in visualizing a Java project**
- Click package explorer in Eclipse
- Choose the project you want to visualize and right click and select “Visualize in SeeIT3D”
- Then within a couple of seconds the visualization as shown above should appear.
- After this you can choose to visualize the whole system such as DrJava for e.g., or just 1 package or even go down to a class, method and line level.
SeeIT 3D showing DrJava system visualized in a container (shown in green wireframe). Each poly cylinder in this case represents the packages. Since it has 29 packages you will see 29 poly cylinders.

The system metric information is visualized at left top side of screen. The rightmost top are the main items available for you to customize the visualization. In the lower section of the SeeIT 3D view are the main interaction options. This panel can be classified into four sections.

- **Granularity Level**: allows choosing the granularity level of the polycylinders contained in the current selection of containers. With the right arrow the granularity will be higher while with the left arrow will be lower.

- **Relationship visual types**: allows selecting the mechanism used to represent relationships between containers; this selection applies to every container selected in the visualization area. You can use the options of Common Base, Arcs, Lines, and Movement.

- **Metric Mapping**: allows changing the mapping between software metrics (LOC, McCabe, LCOM, etc…) and visual properties. This task is accomplished by dragging a certain metric to a specific visual property (Color, Height, or Cross section) or to the Available metrics section to not map it to a visual property. The number you see for the metrics are calculated based on some formula that you do not need to know at this time.

- **Color scales**: allows choosing from a previously defined list of color scales. When a color scale is selected, the visualization is updated and every container in the visualization will be rendered using this color scale.
The table below explains all of the commands that the user can use to analyze and manipulate the graphical elements of the visualization.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
<th>Key binding</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>This option allows visualizing a container from the selected polycylinder. For example, when a polycylinder represents a package, if this action is performed SeeIT 3D will add the corresponding container of the selected polycylinder where the granularity level will be lower than the original container.</td>
<td>Ctrl+E</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>This option allows drawing a rectangle in the visualization area, in order to select multiple polycylinders and containers at once.</td>
<td>S</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>This button will delete the current selected container from the visualization.</td>
<td>Del</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>This option will delete all containers in the visualization.</td>
<td>Shift+Del</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>This action will increase the apparent size of the selected containers.</td>
<td>Alt++</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>This action will decrease the apparent size of the selected containers.</td>
<td>Alt+-</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>This button will toggle the link between visualization and package explorer view. When activated the selection of a polycylinder in the view will trigger the selection of the corresponding artifact in the package explorer view. This way is easy to know what artifact is selected in the SeeIT 3D view.</td>
<td>Ctrl+Alt+S</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>This button will sort the polycylinders of the selected containers in the view. It will take into consideration the visual property selected for sorting i.e. Height or Color.</td>
<td>Ctrl+Alt+S</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>This action will reset the visualization. This means SeeIT 3D will place the containers at the origin of the visualization as well as updating the values of the preferences selected by the user.</td>
<td>Ctrl+R</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>SeeIT 3D allows to save a visualization. Choosing this option will ask for a place to save the current visualization for later loading.</td>
<td>Ctrl+S</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>This option will load a previously saved visualization.</td>
<td>Ctrl+O</td>
</tr>
<tr>
<td>- <img src="image" alt="Icon" /></td>
<td>Make more or less transparent a set of select polycylinders.</td>
<td>Alt+. or Alt+,</td>
</tr>
</tbody>
</table>

**Accessible places in the IDE**

The visualization can be triggered from several places in the IDE. Specifically SeeIT 3D defines three views where a visualization can be triggered:

- Package explorer using the Ctrl+Alt+X key combination or by right clicking the element and selecting the option Visualize in SeeIT 3D,
the Search Results View using the same mechanism as explained above
the Java Editor where right clicking the editor will pop up a menu that allows to visualize
the current Java File, the Parent Package or the Parent Project of the corresponding file.
This figure shows these three options.

Customizable Preferences
SeeIT 3D defines a set of preferences that can be changed by the user in order to feel comfortable
with the tool. These preferences can be done via Windows > Preferences > SeeIT 3D.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Containers per row</td>
<td>When visualizing more than one container, these are place in a table-style location. This way, the containers per row option allows specifying the number of containers that the tool will place per each row</td>
</tr>
<tr>
<td>Polycylinders per row</td>
<td>This option allows specifying the number of polycylinders that will be placed on each row of each container in the visualization</td>
</tr>
<tr>
<td>Background color</td>
<td>This option changes the background color of the visualization area. Changing this option needs to restart the IDE to take effect</td>
</tr>
<tr>
<td>Highlight color</td>
<td>This color indicates the color of the current selection. When a container or polycylinders is selected, it will be drawn using this color</td>
</tr>
<tr>
<td>Relation color</td>
<td>This option changes the color of the representation of the relationships when it applies. For example when visualizing</td>
</tr>
</tbody>
</table>
relationships with Lines/Arcs these will be rendered using this color

<table>
<thead>
<tr>
<th><strong>Scale step</strong></th>
<th>This option is a percentage. It specifies the amount that a container will increase/decrease its size respect to the previous one. This way, if a value of 100 is specified a container will double its size every time the scale up option is selected</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transparency step</strong></td>
<td>The same as the before option, it is a percentage. In this case the value refers to the value of transparency that will be added/removed to a certain set of polycylinders every time the transparency option is invoked</td>
</tr>
<tr>
<td><strong>Color scale by default</strong></td>
<td>The color scale that will be chosen by default when the tool is started</td>
</tr>
</tbody>
</table>
References


75


April 6, 2012

Dr. Bonita Sharif, Principal Investigator  
Ms. Shalini Gadapa, Co-investigator  
Department of Computer Science & Information Systems  
UNIVERSITY

RE: IRB Protocol Number: 140-2012  
Title: Assessing the Usability and Comprehensibility of Software Visualization Tools

Dear Dr. Sharif and Ms. Gadapa:

The Institutional Review Board of Youngstown State University has reviewed the aforementioned Protocol via expedited review, and it has been fully approved.

Any changes in your research activity should be promptly reported to the Institutional Review Board and may not be initiated without IRB approval except where necessary to eliminate hazard to human subjects. Any unanticipated problems involving risks to subjects should also be promptly reported to the IRB. Best wishes in the conduct of your study.

Sincerely,

Peter J. Kasvinsky  
Dean, School of Graduate Studies and Research  
Research Compliance Officer

C: Dr. Kriss Schueller, Chair  
Department of Computer Science & Information Systems