

CoMoVA - A Comprehension Measurement Framework for Visualization Systems

Harkirat Kaur Padda

A Thesis

In the Department

Of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Doctorate in Philosophy (Computer Science) at

Concordia University

Montreal, Quebec, Canada

March 2009

© Harkirat Padda, 2009



Library and Archives
Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file *Votre référence*
ISBN: 978-0-494-63391-5
Our file *Notre référence*
ISBN: 978-0-494-63391-5

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.


Canada

ABSTRACT

CoMoVA – A Comprehension Measurement Framework for Visualization Systems

Harkirat Kaur Padda, Ph.D.
Concordia University, 2009

Despite the burgeoning interest shown in visualizations by many disciplines, there yet remains the unresolved question concerning comprehension. Is the concept that is being communicated through the visual easily grasped and clearly interpreted? Visual comprehension is that characteristic of any visualization system, which deals with how efficiently and effectively users are able to grasp the underlying concepts through suitable interactions provided for exploring the visually represented information. Comprehension has been considered a very complex subject, which is intangible and subjective in nature. Assessment of comprehension can help to determine the true usefulness of visualization systems to the intended users. A principal contribution of this research is the formulation of an empirical evaluation framework for systematically assessing comprehension support provided by a visualization system to its intended users.

To assess comprehension i.e. to measure this seemingly immeasurable factor of visualization systems, we propose a set of criteria based on a detailed analysis of information flow from the raw data to the cognition of information in human mind. Our comprehension criteria are adapted from the pioneering work of two eminent researchers – Donald A. Norman and Aaron Marcus, who have investigated the issues of human perception and cognition, and visual effectiveness respectively. The proposed criteria have been refined with the help of opinions from experts. To gauge and verify the

efficacy of these criteria in a practical sense, they were then applied to a bioinformatics visualization study tool and an immersive art visualization environment.

Given the vast variety of users and their visualization goals, it may be noted that it is difficult for one to decide on the effectiveness of different visualization tools/techniques in a context independent fashion. We therefore propose an innovative way of evaluating a visualization technique by encapsulating it in a visualization pattern where it is seen as a solution to the visualization problem in a specific context. These visualization patterns guide the tool users/evaluators to compare, understand and select appropriate visualization tools/techniques.

Lastly, we propose a novel framework named as CoMoVA (Comprehension Model for Visualization Assessment) that incorporates ‘context of use’, visualization patterns, visual design principles and important cognitive principles into a coherent whole that can be used to effectively tell us in a more quantifiable manner the benefits of visual representations and interactions provided by a system to the intended audience. Our approach of evaluation of visualization systems is similar to other questionnaire-based approaches such as SUMI (Software Usability Measurement Inventory), where all the questions deal with the measurement of a common trait. We apply this framework to two static software visualization tools in the software visualization domain to demonstrate the practical benefits of using such a framework.

ACKNOWLEDGEMENTS

I am indebted to my co-supervisors, Dr. Ahmed Seffah and Dr. Sudhir Mudur, for their helpfulness, guidance and direction throughout my doctoral research. It has been a wonderful academic experience under their supervision.

I am also thankful to Dr. Krishnan and Dr. Olga Ormandjieva for their valuable suggestions that helped me to improve my research work. I would also like to thank Dr. Constantinos Constantinides for his inspirations and kind support.

I am sincerely thankful to all the study participants, practitioners, researchers especially – Dr. Rachid Gherbi, Dr. Abdelwahab Hamou-Lhadj, and usability experts for their kind cooperation. Without them, this research work was not possible.

I would like to express my gratitude to colleagues in our human-centered software engineering group at Concordia, especially – Elaheh Mozzafari, Homa Javahery, Kristina Pitula, Rozita Naghshin, Mohammad Donyaee, and Jonathan Benn for their feedback and motivation. I am also thankful to Yojana Joshi, Daniel Sinnig, Bahman Zamani for their kind support.

Above all, I am grateful to my family. I thank my husband Sukhwinderjit Singh Padda for his moral support and encouragement throughout my PhD work. I am grateful to my daughters Banmeet and Eknoor for giving me hope to achieve something in my life. I am sincerely thankful to my parents and in-laws who gave me continuous inspiration to do research, and maintained my belief of being capable to do it.

Table of Contents

List of Figures	xi
List of Tables	xiii
Chapter 1. Introduction	1
1.1 Motivation.....	2
1.2 Research Statement and Objectives	4
1.3 Methodology	6
1.4 Avenues and Investigations	9
1.5 Challenges Encountered.....	11
1.6 Dissertation Roadmap.....	12
Chapter 2. Visualization Systems: Comprehension of Visual Information	15
2.1 Visualization Systems: Background	16
2.1.1 History.....	16
2.1.2 Definitions.....	16
2.1.3 Classification of Visualization Systems	17
2.1.4 Examples.....	19
2.1.5 General Problems with Visualizations.....	21
2.2 Comprehension: Study Rationale	24
2.3 Comprehension Problems in Visualization Systems	25
2.4 Aspects of Comprehension	26
2.5 Summary.....	33
Chapter 3. Fundamentals of Comprehension Measurement	36
3.1 Comprehension as a Measurable Quality Factor	37
3.1.1 Defining Comprehension	37
3.2 Why Measure Comprehension?.....	39
3.3 Representational Theory of Measurement.....	40
3.3.1 Measurement in Software Engineering.....	42
3.3.2 Measurement Scales.....	48
3.3.3 Data Types and Data Collection.....	50
3.3.4 Evaluation Methods in Software Engineering.....	52

3.4	Measurement of Visualizations.....	54
3.5	Lessons Learned.....	57
Chapter 4. Eliciting Criteria for Comprehension Measurement		59
4.1	Evaluation Foundation	60
4.2	Initial Repository of Comprehension Criteria.....	64
4.3	Refining the Comprehension Criteria	65
4.4	Assessing Completeness by Studying Aspects of Comprehension	67
4.4.1	<i>Categorizing the Comprehension Criteria into Aspects of Comprehension</i> 68	
4.4.2	<i>First Iteration – Aftermaths</i>	86
4.5	Confirming Un-ambiguity, Consistency – Experts’ Opinion	88
4.5.1	<i>Summary of the Experts’ Findings</i>	89
4.5.2	<i>Second Iteration- Aftermaths</i>	93
Chapter 5. Case Studies.....		94
5.1	Verifying Correctness: Case Study 1	95
5.1.1	<i>ADN-Viewer</i>	95
5.1.2	<i>Evaluating ADN-Viewer</i>	97
5.1.3	<i>Experimental Procedure</i>	97
5.1.4	<i>Experimental Results</i>	99
5.2	Verifying Correctness: Case Study 2	101
5.2.1	<i>OSMOSE</i>	102
5.2.2	<i>Evaluating OSMOSE</i>	103
5.2.3	<i>Experimental Procedure</i>	104
5.2.4	<i>Experimental Results</i>	105
5.3	Third Iteration - Aftermaths.....	107
5.4	Making the Criteria Testable	107
Chapter 6. Software Visualization Systems: A Study on Maintenance Tasks		110
6.1	Software Visualizations	112
6.2	Evaluation of Software Visualizations.....	116
6.2.1	<i>Approaches for Empirical Investigations in Software Visualizations</i>	116
6.2.2	<i>Relevant Studies</i>	118

6.3	Identifying the Needs of Software Maintainers	124
6.3.1	<i>Current Work</i>	127
6.3.2	<i>Our Perspective</i>	128
6.4	A Survey-Based Empirical Investigation on Visualization Support for Software Maintenance Activities	135
6.4.1	<i>Survey: Rationale</i>	135
6.4.2	<i>Survey Methodology</i>	135
6.4.3	<i>Discussion</i>	143
6.4.4	<i>Conclusion</i>	146
Chapter 7. Visualization Patterns: A Context-Sensitive Tool to Evaluate Visualization Techniques		
147		
7.1	The Need for Visualization Patterns.....	148
7.2	Patterns Overview.....	151
7.3	Case Study: A Pattern-Oriented Evaluation of Software Visualization Tools	152
7.3.1	<i>Objective</i>	152
7.3.2	<i>Tools/Techniques</i>	153
7.3.3	<i>Software Program for Analysis</i>	155
7.3.4	<i>Tasks</i>	157
7.3.5	<i>Case Study Results</i>	160
7.3.6	<i>Discussion</i>	167
7.3.7	<i>Conclusion</i>	168
Chapter 8. Put It All Together – Comprehension Model for Visualization Assessment (CoMoVA) Framework.....		
169		
8.1	Comprehension: A Working Definition.....	170
8.2	CoMoVA- An Integrated Comprehension Measurement Framework for Visualization Systems.....	174
8.2.1	<i>Activities in CoMoVA</i>	179
8.3	How to Use the Framework?	182
8.4	Conformance to Measurement Theory	184
8.4.1	<i>CoMoVA is a Quality Model</i>	184
8.4.2	<i>CoMoVA and Its' Relationship to ISO 9126</i>	185

8.4.3	<i>Overall Validation Issues</i>	186
Chapter 9.	Operationalization and Overall Validation of the Framework	187
9.1	Measurement Goal Template	188
9.1.1	<i>Context</i>	190
9.2	An Exemplar Study – A Controlled Experiment with Software Visualization Tools 192	
9.2.1	<i>Goals</i>	192
9.2.2	<i>Participants</i>	193
9.2.3	<i>Hypothesis</i>	195
9.2.4	<i>Experimental Variables</i>	196
9.2.5	<i>Types of Experimental Biases and Their Elimination</i>	197
9.2.6	<i>Experimental Setup</i>	198
9.3	Analyzing the Results of a Controlled Experiment	205
9.3.1	<i>Measurement Strategy</i>	206
9.3.2	<i>Confirming the Expertise</i>	212
9.3.3	<i>Analysis of the Gender Differences</i>	213
9.3.4	<i>Validating the Results with Objective Metrics</i>	213
9.3.5	<i>Applicability of Criteria to Visualization Tools</i>	220
9.3.6	<i>Verifying the Null Hypothesis</i>	227
9.4	Discussion and Perspectives	228
Chapter 10.	Conclusions, Contributions and Future Avenues	230
10.1	Concluding Remarks.....	231
10.2	Contributions.....	232
10.3	Research Benefits.....	234
10.4	Future work	236
References		238
Appendix A.	The Proposed Questionnaire	267
A.1	Glossary	267
A.2	The Questionnaire.....	268
Appendix B.	A Survey-based Empirical Investigation for Software Visualization	277
Appendix C.	Proposed Visualization Patterns	282

Appendix D. Participant Evaluation Form	286
D.1 Participant’s Profile	286
D.2 Software Visualization and Maintenance Knowledge	287
D.3 Application Experience.....	287
D.4 Hobbies and Interests.....	287
Appendix E. Informed Consent to Participate in Research.....	288
E.1 Purpose of The Study.....	288
E.2 Procedure	289
E.3 Conditions of Participation	290
Appendix F. Checklist for Study	291
F.1 Before the Test Begins.....	291
F.2 During the Test	292
F.3 After the Test	292
Appendix G. Analysis of Participants’ Responses	293
G.1 Normal Distribution Curves for Radial Technique.....	300
G.2 Normal Distribution Curves for Pyramid Technique.....	305
G.3 Normal Distribution Curves for NestedView Technique	310
G.4 Normal Distribution Curves for Tree Technique.....	315
Appendix H. Analysis of Variance (ANOVA)	320
H.1 One-Way ANOVA Test.....	320
H.2 One Factor Repeated Measure ANOVA.....	324

List of Figures

Figure 2.1: Visualization As a Mapping Process.....	17
Figure 2.2: Numerically Modeled Severe Storm (Wilhelmson et al., 1990).....	19
Figure 2.3: StarTree to Navigate and Explore Hierarchical Relationships (IST, 2005) ...	20
Figure 2.4: Treemap View of Program Execution Data (Orso et al., 2003).....	20
Figure 2.5: Aspects of Comprehension.....	27
Figure 2.6: Gestalt Laws of Visual Perception	31
Figure 2.7: Human Memory System (Gray, 2001).....	32
Figure 2.8: Visualization System and Human Memory Processor	34
Figure 3.1: Understandability and Comprehension	38
Figure 3.2: McCall’ Software Engineering Quality Model	44
Figure 3.3: Boehm’s Model	45
Figure 3.4: GQM Framework for An Organization.....	46
Figure 3.5: ISO 9126 –2000.....	47
Figure 4.1: The Conceptual Models and The Gulfs.....	61
Figure 4.2: Assessment Criteria for Comprehensibility in Visualization Environments (Joshi, 2005).....	65
Figure 4.3: Comprehension Criteria for Visualization Interface Aspect	77
Figure 4.4: Comprehension Criteria for Perception Aspect.....	81
Figure 4.5: Comprehension Criteria for Cognition Aspect.....	85
Figure 4.6: Criteria for Comprehension Assessment in Visualization Systems	86
Figure 4.7: Refined Comprehension Criteria.....	93
Figure 5.1: Screenshots from ADN-Viewer	96
Figure 5.2: Test Protocol for ADN-Viewer	98
Figure 5.3: Screenshots from OSMOSE (Davies, 2004).....	103
Figure 5.4: Test Protocol for OSMOSE.....	104
Figure 6.1: Conceptual Views.....	125
Figure 6.2: Experts’ Categorization of Tasks to Maintenance Activities.....	139
Figure 6.3: Experts’ Opinion on Task Importance	140
Figure 6.4: Intermediates’ Categorization of Tasks to Maintenance Activities	140

Figure 6.5: Intermediates' Opinion on Task Importance.....	141
Figure 6.6: Average Opinion on Visualization Category	142
Figure 6.7: Combined Opinion on Tasks' Categorization.....	143
Figure 6.8: Task Model.....	145
Figure 7.1: Evaluation Strategy for a Technique	149
Figure 7.2: Mapping Technique to Pattern Instances	150
Figure 7.3: Radial Tree Visualization.....	162
Figure 7.4: Pyramid Visualization	164
Figure 7.5: NestedView Visualization.....	165
Figure 7.6: Tree Visualization	166
Figure 8.1: The Proposed Comprehension Framework for Visualization Assessment ..	176
Figure 8.2: The Proposed Comprehension Model for Visualization Assessment (CoMoVA).....	177
Figure 8.3: ISO 9126.....	185
Figure 9.1: Phases of The Experiment.....	204
Figure 9.2: Analysis of Test Session with Morae Manager.....	214
Figure 9.3: Plot of Comprehension Score and Task Time.....	220
Figure 9.4: Rating of Criteria for Each Technique	225
Figure 9.5: Comprehension Score of Each Technique	227

List of Tables

Table 2.1: Scientific Versus Information Visualization (Gershon and Eick, 1997)	18
Table 2.2: Novice Versus Expert Users Needs	23
Table 2.3: Categorizing Comprehension Problems	25
Table 4.1: Properties of Criteria.....	66
Table 5.1: Applicability of Criteria to ADN-Viewer.....	100
Table 5.2: Users' Responses to Criteria.....	101
Table 5.3: Applicability of Criteria to OSMOSE	106
Table 6.1: Types of Strategies (Freimut et al., 2001)	116
Table 6.2: Factors Relating to Choice of Research Technique (Fenton et al., 1997, pp: 120)	117
Table 6.3: Summary Chart of Studies on SV Tools.....	123
Table 6.4: Identified Tasks Along With Their Purpose and Supporting Tools	129
Table 6.5: Classification Results.....	143
Table 7.1: General Format of a Pattern.....	152
Table 7.2: A Pattern-Oriented Analysis of Tools	161
Table 7.3: A Comparative Summary of Tasks.....	167
Table 9.1: Measurement Goal Template (Freimut et al., 2001).....	188
Table 9.2: Installed Hardware and Software.....	192
Table 9.3: Background Variables	194
Table 9.4: Classification of Experimental Biases	197
Table 9.5: Participants' Scores of Comprehension Criteria for Radial Technique.....	209
Table 9.6: Participants' Scores of Comprehension Criteria for Pyramid Technique	210
Table 9.7: Participants' Scores of Comprehension Criteria for NestedView Technique	211
Table 9.8: Participants' Scores of Comprehension Criteria for Tree Technique.....	212
Table 9.9: Comparative Analysis of Techniques.....	215
Table 9.10: Verification of Hypothesis.....	219
Table 9.11: Rating of Criteria for SA4J.....	222
Table 9.12: Rating of Criteria for Creole.....	224

Table H.1: Total Comprehension Score of Each Participant.....	321
Table H.2: ANOVA Results for Radial Technique	322
Table H.3: ANOVA Results for Pyramid Technique	322
Table H.4: ANOVA Results for NestedView Technique.....	323
Table H.5: ANOVA Results for Tree Technique	323
Table H.6: ANOVA Results for Females' and Males' scores	324
Table H.7: Comprehension Score of a Participant for Each Technique.....	325
Table H.8: One Factor Repeated Measures ANOVA Results	326

Chapter 1. Introduction

“Discovery consists of seeing what everybody has seen and thinking what nobody has thought.” – Albert von Szent-Gyorgyi (1893-1986)

Overview

In this thesis, we propose a measurement framework with the principal objective of evaluating the comprehension support provided by the visualization systems to intended users. Towards accomplishing this objective, we have reviewed many different areas in detail – visualization, human computer interaction, and software engineering, and integrated relevant concepts and solutions in the formulation of this measurement framework.

This is an introductory chapter that highlights the problem statement and gives a snapshot of the subject matter presented in this thesis. Here, we present the justification for empirical investigation of visualization systems from a comprehension perspective, which forms the main research pursuit of this thesis. We also describe the research methodology we followed and the investigations we conducted along with various challenges encountered throughout this research. Finally, we give a synopsis of the forthcoming chapters of this thesis.

1.1 Motivation

The value of visuals in a communication process is well recognized starting from the ancient wall paintings to today's computer-based visualizations used in various disciplines. Visualization systems are a form of Human Computer Interaction (HCI), which consist of view/views of data and a suitable interface for interacting with the view(s) (Wilkins, 2003). The potential capability to present huge information in a meaningful and easily perceivable way has resulted in wide promotion of interactive visualizations as providing solutions to this very difficult problem of getting insight into the relationships present in complex and large data in many different domains. This widespread proliferation of visualization tools/techniques in turn highlights the express need for their empirical evaluation. Knight (2001) rightly states that providing evaluations of visualizations is one way to demonstrate that they support a purpose and are adequate for the role claimed for them. The primary role of any visualization is to communicate information using the visual medium, i.e. to portray a set of data in a pictorial form that facilitates its' understanding. Inherent to this portrayal process are constraints in terms of human perceptual and cognitive limitations, physical device screen sizes on which visualizations are displayed etc., that make it harder to comprehend or understand these visualizations. Most existing visualization systems, with their sheer volume of information, place a high cognitive load on the users. It is often unclear as to the extent of help provided by these systems to interpret the meanings of different visuals being displayed.

Kosara et al. (2003) state that no matter how efficient a visualization tool/technique may be, or how well motivated from theory it is, if it does not convey information

effectively then its' usefulness is questionable. Therefore, to assess the empirical evidences of the usefulness/usability of visualization systems from the human point of view, we need to study and answer questions like –

- How well is the visualization system's intent met through visuals and interaction techniques?
- How well is the user's intent met by the visualization system?
- How effective are the visual representations displayed by the system in terms of achieving their major goal of providing 'user insights' for which they were developed? and
- How can we measure whether the visualization has been appropriately comprehended by intended users?

Clearly, a framework that enables us to systematically carry out empirical studies for measuring the comprehension aspects of visualization tools/techniques would be able to provide answers to the above questions. It is important that this framework provides a supporting structure to assess comprehension support provided by the visualization systems in objective terms.

Some evaluation methods have been suggested for visualization systems including empirical assessments with controlled experiments, usability testing and analytical assessments like – heuristic evaluation (Zuk et al., 2006) and cognitive walkthrough (Plaisant, 2004). Despite a growing awareness of the importance of objective evaluation, formal user studies of visualization systems to assess their effectiveness are relatively rare. Unfortunately, performing good user studies is time-consuming and requires substantial expertise in the experiment design and data analysis. In current practice, most

of the empirical studies (Storey et al., 1996; Marcus et al., 2005) with visualization systems are conducted by the original developers/designers and are performed solely for specific objectives. They do not take into consideration the general criteria to assess the effectiveness of these systems. The usability factors in these studies do not cover an important trait of any visualization system i.e. its' ability to ease comprehension of the underlying information depicted through visual(s). Moreover, these methods do not provide significant guidance to assess users' comprehension and are not easy to apply by novices having little knowledge of user interface (UI) practices. On examining the literature (Rushmeier et al., 1995), we found that the software community, especially those working on visualization techniques have also expressed the need for benchmarks or general measurements to evaluate the effectiveness of visualization systems.

Therefore, in our research we are aiming to provide a general measurement framework that could be applied to any visualization system independent of its application area. Furthermore, we propose to quantify the effectiveness of a visualization system in terms of its support for comprehension of the visuals to understand the underlying system. This is made possible through the proposed comprehension criteria and measures that have been investigated thoroughly in this research with a number of usability studies of visualization systems. It is our belief that our research justifies the use of a questionnaire-based evaluation of visualization systems from comprehension viewpoint.

1.2 Research Statement and Objectives

We state our research hypothesis as follows:

Through the use of measurable attributes, the proposed measures-based framework provides significant guidance in the systematic evaluation of comprehension support provided by visualization systems to intended users.

The main objective of this research can now be restated as follows:

- To establish a systematic measurement framework that enables evaluators to assess the available comprehension support provided by a visualization system to the intended audience.

To achieve this objective, we further established secondary objectives as follows:

- To propose a systematic evaluation mechanism that guides the tool users/evaluators to compare and select appropriate visualization tools/techniques. In current literature, the evaluation of visualization techniques is described on an ad-hoc basis, without matching the applicability of techniques to the available context. Towards this endeavour, we propose that every visualization technique may be encapsulated in the form of a visualization pattern describing the applicable ‘context of use’ (i.e. users’, tasks’, and environments’ characteristics) for it.
- To define a suite of criteria to assess the effectiveness of a visualization system in providing user comprehension, preferably independent of the domain, qualitatively or if possible, even quantitatively. Although, researchers in different fields (e.g. psychology, cognition, and HCI) have suggested a few guidelines and principles to follow in the design of visualization tools/techniques, the information seems to be widely scattered and informally defined. Presently, there is no single source that could guide evaluators to determine if the users are able to comprehend the designer’s intentions in the visually represented information along with supported interaction

mechanisms. Our proposed comprehension criteria have resulted from in-depth studies of earlier work addressing psychological, cognitive, and visual communication aspects of a visualization system.

In addition to these two secondary objectives, we also established another research objective for software visualization systems in particular as follows.

- In order to test the applicability and effectiveness of our proposed framework, we opted to use the domain of software visualization. Software visualization systems are developed to ease the comprehension of artifacts comprising the development of large software systems with the goal of providing assistance in software maintenance. To test the effectiveness of software visualization systems, we are proposing an initial catalogue of software maintenance tasks that are purported to be supported by these systems, and are important from the viewpoints of software maintainers to perform maintenance activities. This is required in order to see the functional gaps between the needs of users and the actual tasks supported by currently available software visualization systems.

1.3 Methodology

Our research methodology was composed of the following stages –

- The first stage was to conduct a literature review on visualization systems in order to clearly understand comprehension problems and requirements with visual information, and the need for systematic assessment of comprehension in these systems.
- In the second stage, we conducted further studies of the process of comprehension so as to identify the main aspects involved in the use of any visualization system. We

thoroughly investigated the information flow from the raw data to the cognition of information in human mind. This stage again involved a detailed literature review on comprehension and how we can measure it using the existing measurement approaches and models in software engineering.

- The third stage was to seek important visual design and cognitive principles for effective visual communication to the users. This was based on the work of other renowned HCI researchers who have already investigated the issues of effectiveness for better user experience. Then, based on these principles we proposed a set of comprehension criteria and categorized these measurable criteria into various comprehension aspects explored in the previous stage.
- The fourth stage involved the refinement of these criteria with case studies of visualization systems and opinions of experts as expressed for these criteria. In this phase, we also described the measures for each of the proposed criteria in order to quantify them using a controlled experiment approach. The measures are derived in the form of questions that can be asked to the participants during controlled experimentation with usage of visualization systems.
- The fifth stage was to explore the chosen test domain for our proposed measurement framework i.e. software visualization systems. We performed an exhaustive literature review and conducted an online survey with practitioners and researchers to seek a catalogue of software visualization tasks important from the viewpoints of software maintainers.
- Next, we formulated the process of encapsulating each visualization technique in a pattern format where the context in which the technique can be used is highlighted.

These visualization patterns enable the evaluators and users to compare and understand the functionality of each visualization technique.

- The seventh stage was to formulate our measurement framework using the results of all previous investigations. We have named this framework as Comprehension Model for Visualization Assessment (CoMoVA).
- The last stage encompasses the execution of a controlled experiment with two static software visualization tools. Here, we use our proposed visualization patterns, the catalogue of software comprehension and maintenance tasks, the proposed criteria, and the measures to assess the comprehension support provided by these visualization systems. The data collected from the questionnaires, audio and video recordings are then statistically analyzed using the ANOVA (Analysis of Variance) analysis technique.

Thus our CoMoVA (Comprehension Model for Visualization Assessment) framework incorporates ‘context of use’, visualization patterns, visual design’ principles and important cognitive principles into a coherent whole that an evaluator can use to evaluate the effectiveness of visualization systems. The CoMoVA framework is composed of following components.

Principles – effective visual communication principles (i.e. principle of organization, economization, and communication), and cognitive principles (i.e. principle of naturalness of interaction or mapping, and affordances)

Methods – interviewing technique, online survey, and user studies

Artifacts – questionnaires, repository of comprehension criteria (i.e. Reachability, Simplicity, Clarity, Distinctiveness, Emphasis, Affordance, Dynamism, Appearance, Legibility, Perspective-ness, and Mapping), visualization patterns (for example in the domain of software visualization, we have Radial pattern, Pyramid pattern, NestedView pattern, and Tree pattern), a catalogue of tasks required to be supported by the visualization system (for example, we created a catalogue of 21 software maintenance tasks to be supported by software visualization systems)

Stakeholders – evaluators and/or usability experts, participants

Our set of criteria and measures can be seen as continuity in questionnaire based evaluation approaches, e.g. SUMI (Software Usability Measurement Inventory) which is used to measure user satisfaction based on five usability scales (Kirakowski, 1996). Our criteria can also be reformulated into design principles or heuristics to be tested by experts.

1.4 Avenues and Investigations

A number of avenues have been explored in this research work to achieve the mentioned objectives. These are briefly explained as follows:

1. To refine our proposed comprehension criteria, we met 2 usability experts and sought their valuable opinions for verification of these criteria using an open-ended interviewing technique.
2. To further refine the proposed comprehension criteria, we have conducted two usability studies in two different visualization environments. The first study was conducted with a bioinformatics visualization tool called ‘ADN-Viewer’ (Herisson, 2001). A total of 11 participants from the university community having knowledge of

bioinformatics domain participated in this study. The second usability study was performed with an immersive art visualization environment called ‘OSMOSE’ (Davies, 1996), where 25 participants of varying backgrounds were invited to participate in the study.

3. We have performed an online survey based empirical investigation to categorize the software visualization tasks, as gathered from currently published literature, into traditional maintenance activities. A total of 162 participants were invited worldwide in this investigation. This was done in order to see the effectiveness of software visualization tools in performing the maintenance activities. Through this survey, we proposed a catalogue of the software comprehension and maintenance tasks that are required to be supported by current software visualization tools. This initial repository of 21 software comprehension and maintenance tasks can provide guidance in evaluating software visualization tools from ‘functional’ viewpoint i.e. the evaluator can determine which of the tasks from this repository are supported by any software visualization tool in hand.
4. To illustrate the usage of our framework for visualization systems, we conducted a controlled experiment in our human-centered software engineering lab with two static software visualization tools i.e. ‘Structural Analysis for Java (SA4J)’ (Iskold et al., 2004) and ‘Creole’ (Callendar, 2006). 15 participants from the university community having knowledge of software maintenance and visualizations in general participated in the experiment.

1.5 Challenges Encountered

Through out in this research we were trying to measure what would normally be considered intangible, and regarded usually as immeasurable by the researchers, namely comprehension support. This has been a very challenging task by itself as a number of complex issues are involved. Three main human senses (i.e. sight, touch and hearing) are involved in making sense of the multi-media visualization(s), which further complicates the problem of comprehension assessment. Therefore, we simply restricted ourselves to the 'sight' sense in this research. Moreover, we cannot directly look inside the mind of a person to guess what he/she is thinking about any visualization. Therefore, we adopted an indirect approach where we investigated the tangible properties of the visualization systems and saw their impact on the performance of an individual. A number of hurdles have been encountered throughout this research.

1. The very first problem was the verification of proposed criteria. There is no scientific method to apply for verification, and therefore we sought the opinions of experts in relevant fields. However, asking the experts to comment on their judgment was also not easy.
2. Visualizations are employed in a number of different domains and therefore our second challenge dealt with selecting a suitable application area to apply our proposed framework. We selected software visualizations as our application area because of our background in software engineering field.
3. The third barrier was the selection of appropriate tools for study purposes. We found that most of the software visualization tools are research prototypes and are not fully functional. We selected only two static software visualization tools for our study.

4. The fourth obstacle in our research was that of selecting suitable participants for study purposes. During our three in-lab studies and one online survey, we have found that it is not easy to get people to commit their valued time towards participating in the usability experiment/survey and performing the assigned tasks.

1.6 Dissertation Roadmap

A brief explanation of the remainder of this dissertation is as follows.

- In **chapter 2**, we discuss the background on visualization systems with a focus on the concerns for comprehension assessment. In particular, this chapter highlights the need to evaluate visualization systems from the viewpoint of comprehension aspects involved in the interaction with these systems, and forms the justification for our proposed comprehension measurement framework.
- In **chapter 3**, we present the fundamentals of comprehension measurement. In order to provide a foundation work for the establishment of our framework, we further study comprehension and explore the current state of art in software measurement, existing measurement strategies along with related measurement studies of visualization systems previously reported.
- In **chapter 4**, we propose a set of criteria based on cognitive, perceptual, and visual interface properties of visualization systems. The criteria introduced in this chapter are verified for their completeness, consistency, and un-ambiguity properties. A number of methods, like – conducting a comprehensive literature review and open-ended interview with experts, have been conducted to propose a minimal set of comprehension criteria.

- In **chapter 5**, we conduct two case studies to test the applicability and effectiveness of our proposed criteria. The first case study is performed with a bioinformatics visualization tool to evaluate its' effectiveness based on the proposed comprehension criteria. The second case study is conducted in immersive art visualization environment. Both these case studies are analyzed thoroughly in this chapter along with a final set of measures to assess the proposed comprehension criteria.
- In **chapter 6**, we perform an in-depth investigation of our application area i.e. software visualization systems. This chapter discusses the related studies on the empirical investigation of software visualization systems along with our online survey-based empirical investigation of these systems with an objective to identify the gap between the needs of users and the tasks supported by current software visualization systems.
- In **chapter 7**, we formulate and describe the process of encapsulating a visualization technique in the form of a visualization pattern where the 'context of use' in which the technique is applicable is appropriately summed up to capture the boundaries of evaluation. We demonstrate this by devising a set of four visualization patterns for the four different visualization techniques employed in two software visualization tools under our investigation.
- In **chapter 8**, we integrate all the knowledge gained from previous chapters to present our comprehension measurement framework. This chapter describes the components and structure of our proposed Comprehension Model for Visualization Assessment (CoMoVA) framework.

- In **chapter 9**, we test our proposed measurement framework in software visualization application area using a controlled experiment approach with two static software visualization tools. Here, we analyze the results to assess the comprehension support provided by software visualization systems under our study using the comprehension criteria in our framework.
- Finally **chapter 10** summarizes the work, major research contributions that have resulted, research benefits that can be reaped from our measurement framework, and future avenues for research in this area.
- In addition, a number of appendices are also included which provide details about the online survey, visualization patterns, the consent and user evaluation forms, the questionnaire, and more details of ANOVA analysis of the controlled experiment.

Chapter 2. Visualization Systems: Comprehension of Visual Information

“Providing evaluations of visualisations is one way to demonstrate that they support a purpose and are adequate for the role claimed for them....” – (Claire Knight, 2001)

Overview

In this chapter, firstly we present our background study on visualization systems and the general problems with them that highlight the need for evaluation of these systems. Secondly, we provide our justification for the proposed comprehension measurement framework in order to evaluate the visualization systems. Towards this main objective, we also outline the specific comprehension problems and the aspects involved in comprehending the visual information in visualization systems. We study the information flow from rendered data to the cognition of information in human mind with an objective to understand the aspects involved in the process of comprehension of visual information presented by visualization systems. These are aspects that significantly affect overall user understanding of displayed visual information.

2.1 Visualization Systems: Background

Before addressing the research question, on why we need to have a comprehension measurement framework for visualization systems, a brief background on the subject of visualization is presented here. This section introduces the evolution of computer-generated visualization, its application areas along with some illustrative examples, as well as the associated potential pitfalls.

2.1.1 History

The pedigree of visualization has its roots in pictorial representations dating back to the origins of man when pictographs or man made images were used for communication. “Through the centuries, we have seen human generated maps of the sections of the world for travel and warfare, images of the positions of stars and other celestial bodies, imagery of plans for architectural and novel devices, images to enhance stories, and many more such examples. These early steps comprise the beginnings of the husbandry of visualizations. To support many of modern endeavours, computer generated data visualizations called ‘plots’ appeared in the late 1940’s, when tables became too large for a human to comprehend and manage” (Baker et al., 2005). These visualizations were followed by the growth of computer graphics and systems that permitted the rapid, often interactive generation of scientific data sets. With the prosperity of visualization in scientific computing, professionals from other disciplines like statistics also began using computer-based visualizations to support their data exploration tasks.

2.1.2 Definitions

In the general context, the term “to visualize” is defined in Oxford English Dictionary as – “to form a mental vision, image, or picture of (something not visible or present to the

sight or of an abstraction); to make visible to the mind or imagination.” (OED, 2005) Visualization is an effective way to communicate abstract as well as concrete ideas through visual imagery.

According to Foley and Ribarsky (1994), “a useful definition of visualization might be the binding (or mapping) of data to a representation that can be perceived. The types of binding could be visual, auditory, tactile, etc. or a combination of these”. Visualization can be also be seen as "a computer generated image or collection of images, possibly ordered, using a computer representation of data as its primary source and a human as its primary target" (Baker at al., 2005). Visualization is a mapping process from computer representations to perceptual representations, choosing encoding techniques to maximize human understanding and communication (Owen, 1999) as shown in Figure 2.1.

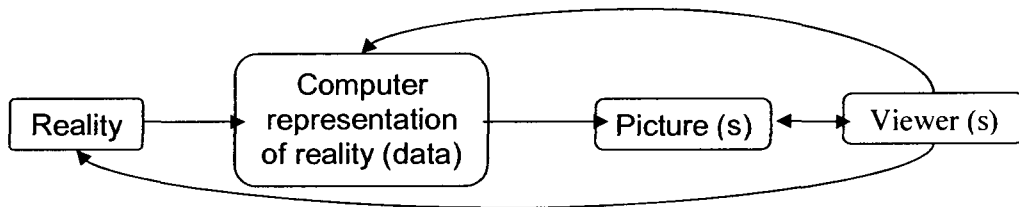


Figure 2.1: Visualization As a Mapping Process

The back arrows in Figure 2.1 depict that viewer(s) may view visual(s) to get a deeper understanding of the reality or the mathematical concepts, and/or to get a visual proof of computer representations derived for physical phenomena or concepts.

2.1.3 Classification of Visualization Systems

As the need and opportunities grew with the advancement of computer technologies, researchers have shown their burgeoning interest in computer-based visualizations and have classified them accordingly. In general, there are two main forms of visualizations i.e. scientific visualization and information visualization.

- Scientific visualization mainly deals with visual representation of scientific data to explore and understand natural phenomenon(s). This data is captured from a physics-based model(s).
- Information visualization, as opposed to scientific visualization, aims to visually present abstract data that may have no natural visual representation. This data can be very complex, containing a large number of elements, structured hierarchically in a network, linearly, or could even lack any kind of structure.

Comparison of information visualization with scientific visualization in terms of intended audience, task, input data and input size is shown in Table 2.1.

Table 2.1: Scientific Versus Information Visualization (Gershon and Eick, 1997)

Visualization Type	Audience	Task	Input	Input Quantity
Scientific Visualization	Specialized, highly technical	Deep understanding of scientific phenomena	Physical data, measurements, simulation output	Small to massive
Information Visualization	Diverse, widespread, less technical	Searching, discovering relationships, including action (fast, many times!)	Relationships, nonphysical data, information	Small to massive

These two categories are further classified by visualization researchers according to the application areas where the visualizations are applied. Therefore, today we are presented with a broader context of visualizations named according to application

domains like – database visualization, software visualization, biomedical and geospatial visualization and so on.

2.1.4 Examples

Below, we discuss a few of many examples available to illustrate the power of visualization in gaining insights and understandings into complex data or artifacts.

a) *Scientific visualization*

In scientific visualization, the goal is typically to visualize scientific phenomena from data experimentally captured or through simulation programs. Figure 2.2 shows the famous “visualization of a storm” employed in geophysics, where it has been used by environmental scientists to study the storm phenomenon.

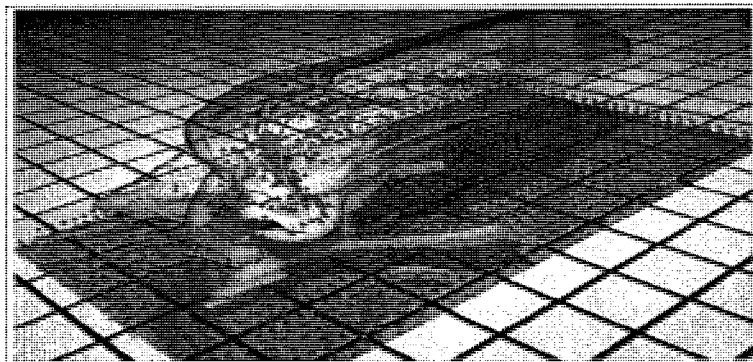


Figure 2.2: Numerically Modeled Severe Storm (Wilhelmson et al., 1990)

b) *Information visualization*

In information visualization, one is typically looking for complex relationships that are not obvious from non-pictorial representations. Figure 2.3 illustrates a hyperbolic view of a complex hierarchy.

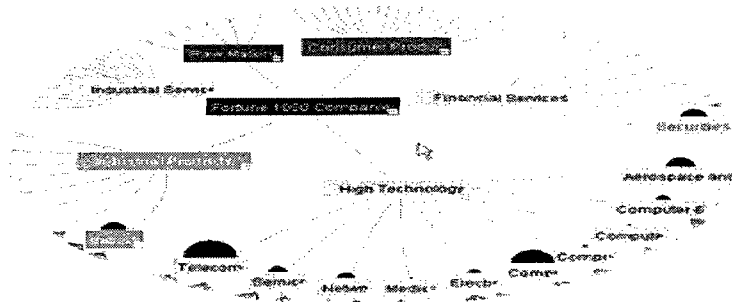


Figure 2.3: StarTree to Navigate and Explore Hierarchical Relationships (IST, 2005)

c) Software visualization

Software visualization can be viewed as a specialized subset of information visualization that uses visual representations to make software more visible. This is because information visualization is the process of creating a graphical representation of the abstract data and this is what is required when we try to visualize software components (Knight and Munro, 2001). Software is inherently complex having a large number of artifacts in the system and their relationships, so we need to visualize software in order to comprehend the meaning of these artifacts. Software visualization is concerned with the construction of static and dynamic views of the software systems. For example, Figure 2.4 depicts the program execution data using a Treemap technique.

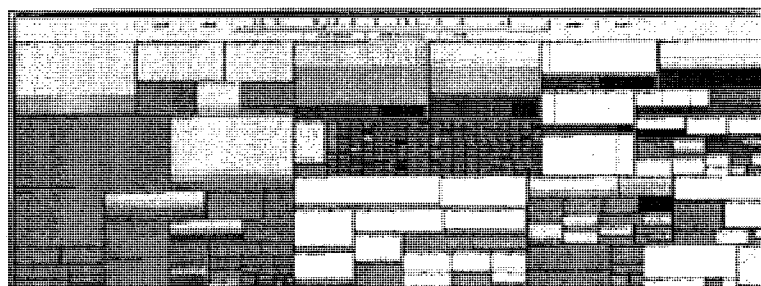


Figure 2.4: Treemap View of Program Execution Data (Orso et al., 2003)

2.1.5 General Problems with Visualizations

Despite the application of visualization in diverse fields as illustrated above, various researchers have come across many problems with them. Based on the literature survey, we are summarizing some potential pitfalls associated with visualizations that are hindering their usefulness to ultimate users.

a. Cognitive overload

First and foremost, we need to address the basic question of ‘user insights’ for which visualizations were developed. Due to large amount of information that is displayed in visualizations they are becoming overly complex, thus burdening users’ minds with information, commonly referred to as the information overload problem. Pfitzner et al. (2003) state that a higher cognitive load is placed upon the user if the visualizations are difficult to interpret.

b. Flashy imagery

Ma (2004) points out that many research visualizations tend toward colourful, showy images rather than informative ones. He says that ineffective visualizations are typically caused by the careless use of visual metaphors, a rush to publication with immature research results or a desire to generate eye-catching images for a publication. Although fascination attracts the audience, it is only temporary and finally users prefer the visualization based on its’ inherent content and usability.

c. Lack of evaluation

While tremendous advances have been made in the field of visual rendering, the growth of usability studies and empirical evaluations has been relatively slow (Ma, 2004; Chen 2005). According to Ma (2004) many visualization research results are

mainly good for publications and demonstrations but are not directly applicable in a real-world problem-solving environment, and this is because the knowledge of application scientists is not fed into the visualization tools/techniques by visualization researchers. Moreover, scientists who are expert in their field do not like to use the visualizations created by novices (Petre et al., 1998). To deliver truly usable visualization solutions, we need to measure the effectiveness of the visualization methods. User study should be added to the introduction of every new visualization technique to assess the real context in which it is useful (Ma, 2004). Usability studies need to address whether users can recognize the intended patterns being presented through visualizations (Chen, 2005).

d. Lack of scalability

Nowadays, another major bottleneck that hinders the making of good visualizations is the sheer volume of data coming from scientific sources. Many of the present techniques do not scale with the problem size. We need strategies to organize and operate on data providing the desired interactivity and display resolution, and with available computing resources (Ma, 2004).

e. Navigation problems

Non-intuitive navigation is a factor that frustrates the users most while exploring the visualization environment. Many researchers have noted that visualization environments are difficult to navigate, and are sometimes even more so when it comes to interpretation of the results. Furthermore, many of the visualizations do not offer guidance for ‘where to look’ and ‘what to look for’ during the exploration (Bramer et al., 2002).

f. Improper context of use

Generally, most of the visualizations lack the proper guidance on the real context in which the tool or technique is applicable. There is always a gap between the novice and expert user's knowledge; novice users normally do not possess the same analytic abilities as experts, which may hinder their ability to interpret visualizations (Bramer et al., 2002). Moreover, their needs are not the same, so they have their own objectives. Table 2.2 highlights some of the varying needs of expert and novice users while exploring any visualization system.

Table 2.2: Novice Versus Expert Users Needs

Novice Users	Expert Users
<input type="checkbox"/> Need a visualization system that is very easy to use.	<input type="checkbox"/> Need a system that complements and supplements their thinking.
<input type="checkbox"/> Need clear and detailed help.	<input type="checkbox"/> Need a very flexible system, allowing seeing different levels of details.
<input type="checkbox"/> Do not need many different views of visualization.	

Many tools and techniques are developed without taking into consideration the environment in which they will be effective and as a result users falsely assume that they are universally applicable and then become discouraged with their real use.

g. Interaction difficulty

“A crucial factor in the usefulness of a visualisation system is the ease with which the analyst can interact with the visualisation to obtain the information they require” (Pacione, 2004).

2.2 Comprehension: Study Rationale

In the previous section, we have seen a number of problems that could negatively affect the overall value/quality of visualizations. Researchers (Garvin, 1984) and standards (ISO 9126, 1991) have included “user view” or “user perspective” as one of the facets of quality of any software product. To assess the “quality in use” or the user’s perspective of the quality of any visualization system, we need to recognize the factors that affect it. Based on users’ viewpoint, when we look at the visualization systems the most predominant factor that affects their ultimate quality is user comprehension of the visual information presented. The success of any visualization system relies on its support for providing ‘user insights’ to understand underlying artifacts represented through the visual. If the visualization system does not achieve this objective, it is of little use or suffers from poor quality. Thus, comprehension of the visual information presented by the system is the most important feature to determine the quality of any visualization system.

The justification for this statement, based on the comments of various researchers found during the literature survey, is as follows:

- Saltz and Steinbach (1997) suggest that the innovative display of the information, by itself, is not enough to ensure the success of the visualization system, as the users must intuitively understand the visualization that has been created.
- Cross II et al. (1999) state that visualization in and of itself, however, is not necessarily beneficial. There are many concerns including the cognitive issues relating to the user and the process of human comprehension that influence the utility of visualizations.

- Friendly (1999) says "Like poor writing, bad graphical displays distort or obscure the data, making it harder to understand or compare."

Having established a consensus on importance of this characteristic of any visualization system, we further studied it in order to categorize the specific comprehension problems that users may encounter in any visualization environment as discussed further in the following section.

2.3 Comprehension Problems in Visualization Systems

Ekenstierna (2002) has defined a help question model for various user assistance techniques based on the probing questions that arise in users' mind to make a mental model of the system like – what, how, why, when, where etc. This classification is analogous to what the users think about when they interact or explore any visualization system. Based on this model we can identify five different comprehension problems in visualization systems as shown in Table 2.3.

Table 2.3: Categorizing Comprehension Problems

Problem Type	Problem Description
Goal-oriented	What can I do with this visualization?
Descriptive	What is this? What does this do?
Procedural	How do I do this?
Interpretive	Why did it happen? What does it mean? When is it appropriate?
Navigational	Where am I? Which path to follow to go from this position to that?

A brief description of these problems is as under:

- Goal-oriented problems are basically the first comprehension problems that come up when a user starts looking at any visualization system. The user starts thinking of the overall objective or goal of using the visualization system. The other comprehension problems start arising only when the user is going to pay more attention to the problem context displayed in the visualization.
- Descriptive problems are the knowledge acquaintance problems. After the initial sensing, now the users' working memory starts asking questions to gain an understanding of the problem domain represented in visualization.
- Procedural problems deal with investigation of systematic procedures to achieve a particular user goal. These generally address the strategies to be adopted while accomplishing the goals with any visualization system.
- Interpretive problems are the kind of user comprehension tasks where users make use of their long-term memory to analyze the current situations. The users make use of their past knowledge as benchmark data for interpretation of present knowledge.
- Navigational problems are mainly dealing with finding the pathways in complex and large information spaces represented in visualization systems.

In order to better understand these comprehension problems highlighted by the review, we further studied the process of comprehending visual information and identified various aspects involved in it as presented in the next section.

2.4 Aspects of Comprehension

The notion of user' comprehension assessment is not an easy task, as there are many contributing aspects involved in it. To capture these aspects, we studied thoroughly the

communication path starting from data or information that is rendered in visual form to the perception and cognition of information in human mind. For any visualization system, data rendered in visual form is perceived or interacted upon by the user of that system as shown in Figure 2.5.

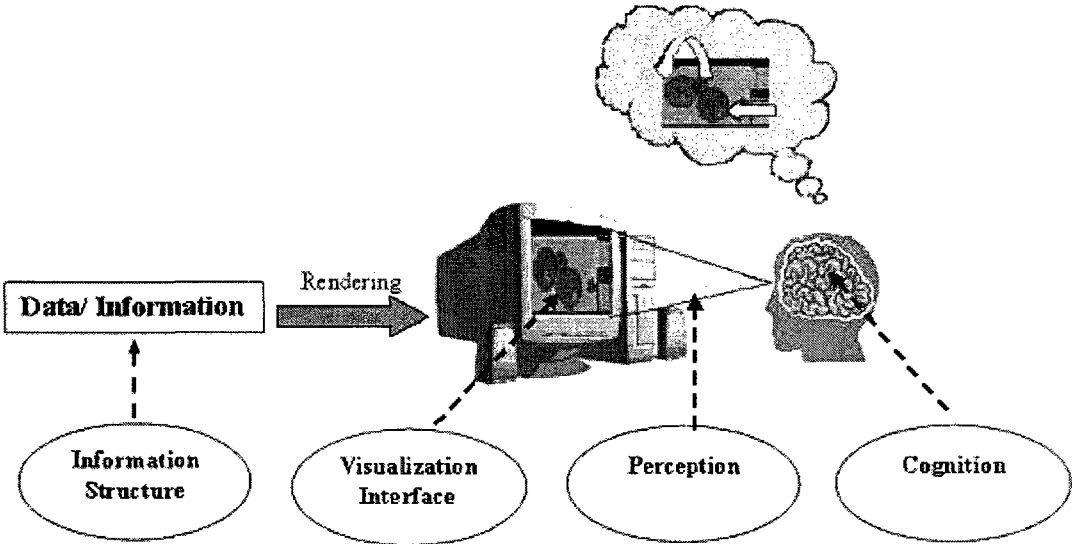


Figure 2.5: Aspects of Comprehension

As mentioned previously, in this information flow starting from raw data to the cognition of information in human mind for comprehension, we believe there are “aspects” which play significant roles and affect one another. We term these aspects as – “Information Structure”, “Visualization Interface”, “Perception”, and “Cognition” as shown in Figure 2.5. A detailed explanation of each of these aspects is as under.

▪ **Information Structure**

The information structure has a profound affect on user comprehension. Differences between users' expectations and the actual information structure may cause comprehension difficulties. Reliability of the data is affected by the nature of gathering or processing data causing noise to be added to the original data, as well as visualization

constraints which result in changing the original information in order to adapt it to the particular technique (Luzzardi et al., 2004). Gershon (1998) states that flaws in the data reduce the accuracy and possible usefulness of the resulting visualization. Sometimes the data that is rendered is not perfect by itself due to many causes like – corruption of data, incompleteness, inconsistency, information complexity, uncertainty, imperfect presentation etc (Gershon, 1998). The net affect is that the visual used to represent the data/information does not represent the whole story and is not easy to comprehend. Luzzardi et al. (2004) and Brath (1997) have suggested some measures to apply for information complexity like – data density, data dimension etc. These measures of information structure are beyond the scope of our research and are not studied in detail further. In our research, we believe that the data/information that is rendered is free from the kind of flaws mentioned above by Gershon (1998).

Once the data is visualized, it is presented to the user on screen. So, the next aspect to consider for user comprehension is visualization itself which includes the view of the data in the form of a visual and the user interface for its' manipulation.

- **Visualization Interface**

According to Wilkins (2003), the visualization presented to the user consists of two parts – a view of the data and a graphical user interface (GUI) associated with the view. The view is a representation of the data that is derived from various data features and task requirements. Each view has its own intent that captures the general purpose and motivation leading to its design (Storey et al., 2005). For example, a graph typically has the intent of showing trend, a tree shows hierarchy, a graph shows connectivity etc. The GUI augments the view and usually consists of standard graphical components like –

menus, buttons, sliders, list boxes, etc. User interacts not only with the view but also with the GUI to achieve user goals. The interface is a crucial part of any visualization system, as it essentially forms the link between the user and the visualization itself. An easily understandable UI helps the user to interpret the visualization and perform correct operations. So, in order to comprehend the visualization accurately, we should explore the view and its accompanying interface.

The next aspect to consider for user comprehension is perception.

- **Perception**

Perception is an integral part of any visualization and details that can not be perceived by the observer serves no purpose if displayed (Kjelldahl, 2003). As information contained in visual must pass through the perceptual system, therefore effectiveness of visual also depends on their perceptual characteristics (Rheingans and Landreth, 1995). In psychology and cognitive sciences, perception is the process of acquiring, interpreting, selecting, and organizing sensory information (Wikipedia, 2007). There are five classical senses – sight, hearing, touch, smell, and taste. Each of these senses plays a significant role in perceiving the information around us. In our research, we limit our scope to the study of factors in visualization systems that affect the “sight sense” or the “vision capability” of the users. There are various perceptual attributes of visuals, like – color, line orientation, contrast, transparency, position and size etc., that make them easier to comprehend by eyes. Lowe (1999 and 2003) has conducted studies on visualization and perception, and has shown that perceptual features of a visual can interfere with successful comprehension. According to Wünsche and Lobb (2001) perception of a scene is processed in two stages: pre-attentive and focused attention stage. They state that the

pre-attentive stage allows perception of very simple primitive textual features, like – length, width, orientations and interactions along with shape, color, intensity, texture depth etc., without conscious attention. This initial stage is followed by focused attention stage, which entails conscious examination of a scene, rapid mental calculations and quantitative reasoning for complex information objects. Schiffman (1996) suggests that perception can also be dependent on previous stimuli, and familiar shapes and configurations can improve recognition of a target. To utilize the strengths of human visual system and to reduce cognitive load there are a set of basic organizing principles called Gestalt Laws. The Gestalt approach emphasizes that we perceive visual components as well-organized patterns rather than separate components. Gestalt is a German word that translates to "configuration or pattern". According to Gestalt theory, there are six main laws that determine how we group things according to visual perception, these are – Proximity, Similarity, Closure, Symmetry, Common fate and Continuity (c.f. Figure 2.6). Each of these laws describes the strengths of human visual system in perceiving visual objects. A brief explanation of these laws is as follows:

- The law of proximity states that objects that are close together will tend to be perceived as a group.
- In the same way, similarity law states that objects of similar physical attributes like – shape, size, color etc. tend to be grouped together.
- The principle of continuity states that continuous forms are more likely to dominate a scene in comparison to forms that have abrupt changes in direction, i.e. objects that lie along a common line or curve tend to be grouped together.

- Connectedness, which is a form of continuity states that connected objects are perceived as groups.
- Closure is the form of common enclosed region.
- Principle of common fate states that objects that have the same orientation or motion are also grouped together.

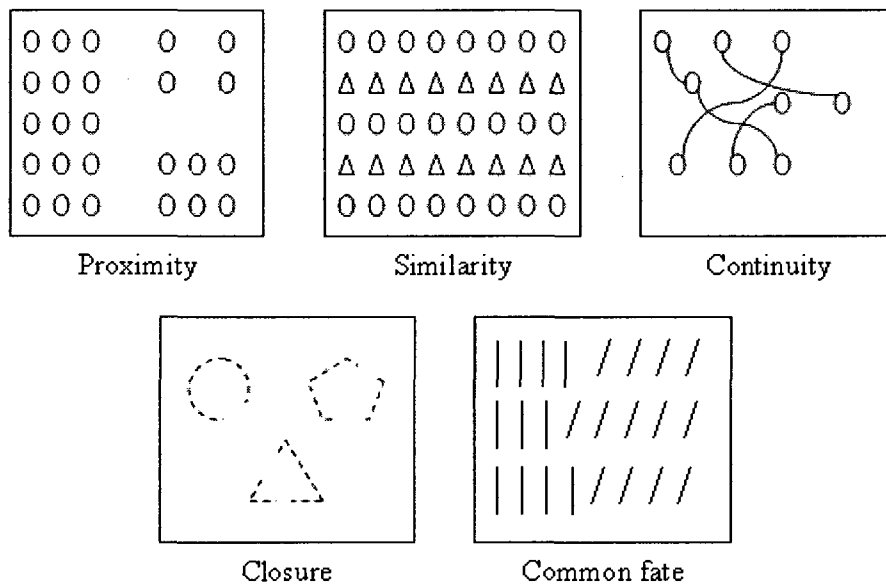


Figure 2.6: Gestalt Laws of Visual Perception

The final aspect to consider for comprehension is cognition.

▪ **Cognition**

In order to judge the degree of comprehension, it is also necessary to understand human information processing or the cognition of information in human mind. This is important because humans have limited information processing capacity. The classic model of human memory system shown in Figure 2.7 is composed of three major components: sensory memory storage, short-term or working memory, and long-term memory. In this information-processing model of mind, sensory information enters ‘sensory storage’ which behaves like an input buffer in a computer.

Once in the sensory storage, the information is either passed to short-term memory component by the attentional mechanism, or it is lost, i.e. being “written over” or “masked by” successive information or “decays” (in approximately 200-250 milliseconds for the visual sensory memory, or iconic storage, and approximately 4-7 seconds for the auditory sensory memory, or echoic storage) if it is not refreshed (Hewett, 2003). When the information is selected for further processing then it is passed to short-term memory.

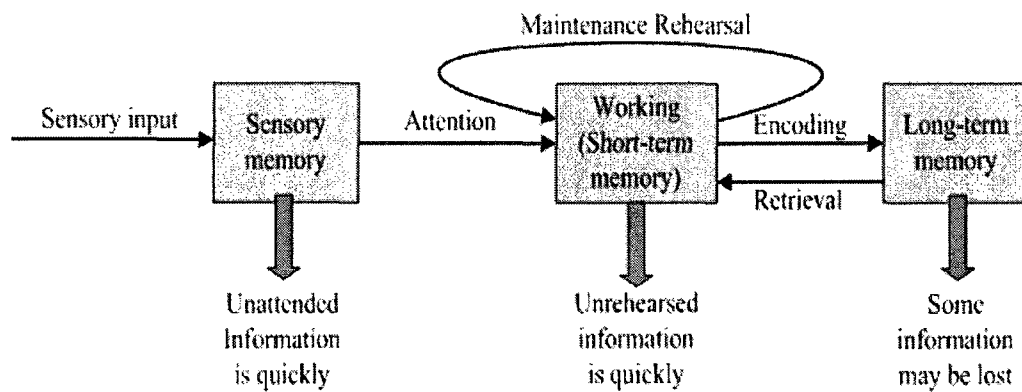


Figure 2.7: Human Memory System (Gray, 2001)

Short-term or working memory (STM) is a buffer where concepts are stored during the initial stage of comprehension. STM limitations vary depending on the individual and on what kind of information is being retained (Kintsch, 1998). According to Hewett (2003) – “STM is also described as having a limited storage capacity (seven plus or minus two chunks) for a relatively brief duration (estimates range from 12 to 30 seconds without rehearsal) before information is lost through simple decay or when new information displaces the older information; however, information can be maintained in it for periods of time longer than 20 seconds with maintenance rehearsal.” Kintsch and Dijk (1978) claim that the risk of comprehension errors increases with the density of information stored in STM. The information is finally encoded into Long-term memory

(LTM), where information is represented as concepts and associations between concepts is presented through schemas or patterns. The retrieval performance of LTM depends on the density of associations between these concepts (Anders and Kintsch, 1995). Several types of information are encoded in LTM, including things like – facts and events, motor and perceptual skills, knowledge of physical laws and systems of mathematics, a spatial model of the world around us, attitudes and beliefs about ourselves and others etc. (Hewett, 2003). Practically, LTM is considered as unlimited in capacity. However, it also fades over time. A list of general concepts, once remembered, deteriorates to a level of about 60% after 3 months and stabilizes at around 25% after 3 years (Reed, 1996).

In short, from above explanations, we conclude that cognition being an important component of the comprehension process is a complex aspect by itself. Therefore, direct assessment of each user's cognition is a problematic task as users have varying cognitive qualities, which are also impacted by several physical, social, and environmental factors. The study of these external factors is beyond the scope of this research. So, in order to measure each user's cognitive aspect of comprehension, we simply observe those factors in the visualizations systems that impact the cognitive capabilities of the users.

2.5 Summary

As we have seen in the previous section, grasping information from the visualization and interpreting them mentally is a complex process that involves a number of different aspects. We also observed that direct assessment of comprehension is not feasible as these aspects are interrelated and affect one another to make a mental model of any data represented through visualization(s). Therefore, to measure this seemingly immeasurable characteristic of visualization systems, we are limiting our scope as discussed further.

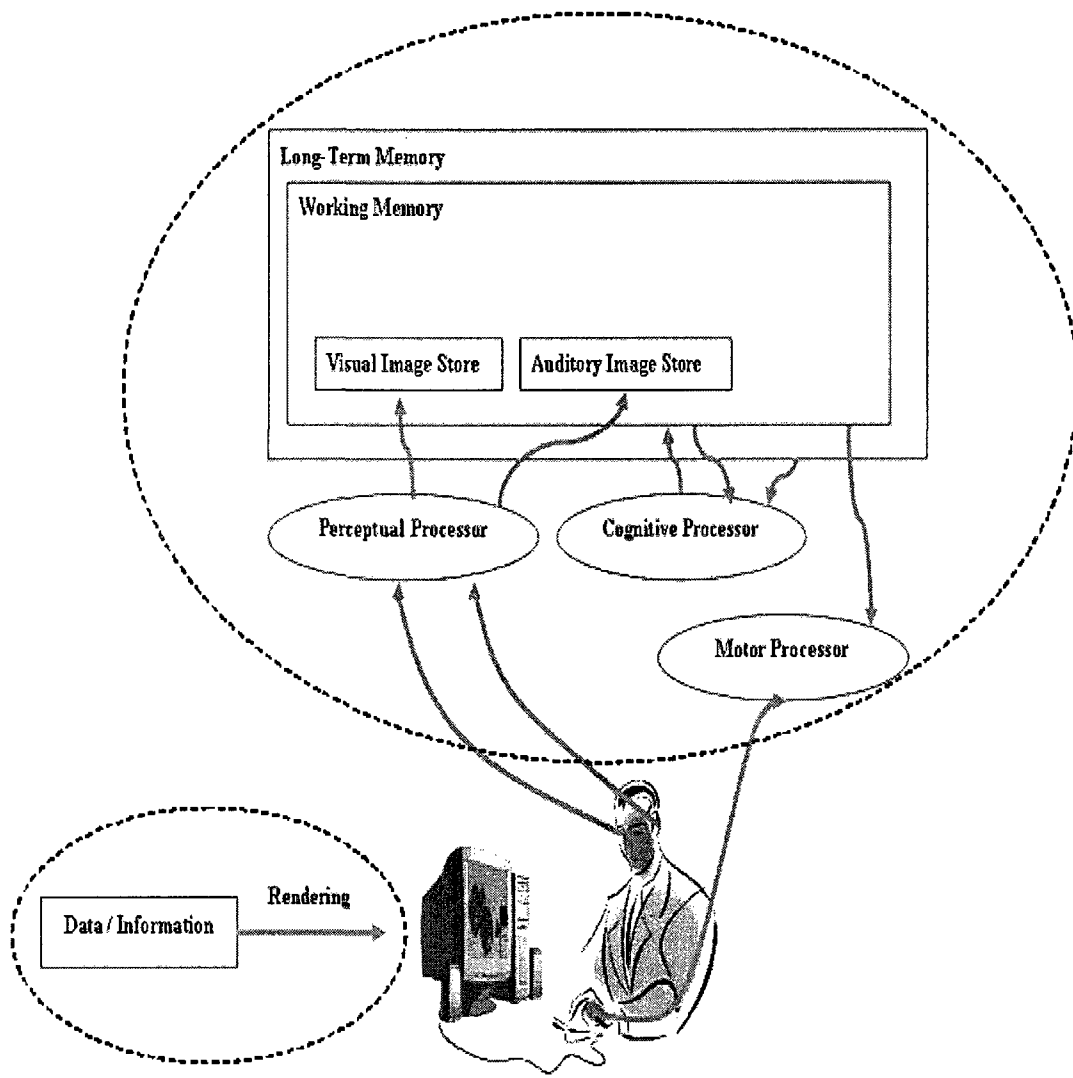


Figure 2.8: Visualization System and Human Memory Processor

In Figure 2.8, the large dashed oval depicts the model of human processor as proposed by Card et al. (1983) and small dashed oval illustrates the “Information Structure” aspect of comprehension. In the model of human processor, three inputs (i.e. - visual, audio and movement) into human brain are processed by three different internal processors. Although, both of these concepts represented in dashed ovals influence the comprehensibility of the visualization systems, the direct measurement of their impact on comprehension is not possible. Therefore, in our research we just look at those visible

features in any visualization system that contribute to these abstract concepts and could guide us to indirectly measure the comprehensibility of any user. To measure comprehension, we study the “Visualization Interface” aspect in detail along with those features in the visualizations that affect the “Perception” and “Cognition” aspects of comprehension. We study these aspects further in order to determine the criteria affecting user comprehension of a visualization system. The audible input to the perceptual processor of the human processor model as shown in Figure 2.8 is also not part of the scope of this research, as we are limiting ourselves only to the visual attributes of the visualization system.

Chapter 3. Fundamentals of Comprehension

Measurement

“...when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind....” – Lord Kelvin (1824-1907)

Overview

This chapter deals with the basics of measurement in general. It begins with our requirement that comprehension should be a measurable quality factor of visualization systems. It then discusses the current state of art in measurement by elaborating various issues pertaining to the measurement process like – measurement models, measurement scales, data collection procedures, and evaluation methods. Related studies on measurement in visualizations by other researchers are also discussed here. Finally, distilling knowledge from the above, we present our concluding remarks for the establishment of our framework.

3.1 Comprehension as a Measurable Quality Factor

We consider comprehension of visual information presented as a fundamental characteristic that influences the overall quality of any visualization system. Being a quality factor, we believe that it must be decomposable into measurable attributes that can be measured using some measurement scheme.

In order to further proceed with its measurement, we first describe how various researchers perceive comprehension in the context of software and information visualization.

3.1.1 Defining Comprehension

In simple terms, comprehension refers to activities that humans do: understanding, conceptualizing, and reasoning about the artifact under consideration. Klemola and Rilling (2003) state that comprehension consists of several processes including – recognition, learning, grouping concepts or chunking, searching for occurrences of a term or tracing, and depends on the familiarity of an individual with the artifact in question. In another paper (Klemola and Rilling, 2002), these authors have identified a hierarchy of five comprehension tasks, which are –

- the recognition of a familiar term,
- the tracing of references to a term,
- the memorizing of new information,
- the learning of information, and
- the creation of information.

Many other researchers have defined the term ‘comprehension’ as follows:

- In cognitive science, comprehension is often characterized as the construction of a mental model that represents the objects and semantic relations described in a text (Kintsch and Dijk, 1978).
- It is a constructive process in which an individual uses prior knowledge, information presented in the external media, and skills of reasoning and mental visualization to build a mental model of the system (Narayan, 1997).

Comprehension is often confused with understandability. However, according to Cioch (1991), understandability consists of two components: comprehension and lack of misinterpretation as shown in Figure 3.1.

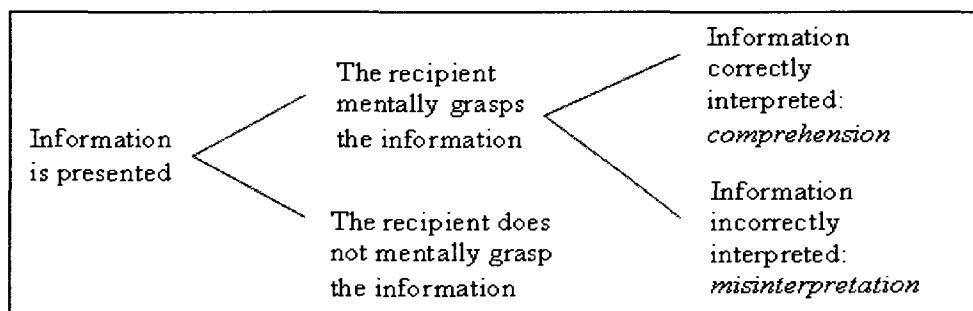


Figure 3.1: Understandability and Comprehension

“When one wishes to ascertain the understandability of a particular software-related product, one is often concerned not only with the degree to which, or the ease with which, the information is grasped mentally, but also with the degree to which it is misinterpreted by the person examining the product” (Cioch, 1991). So, comprehension is one aspect of understandability, which means that the person is able to mentally grasp the information and to interpret it correctly. Lack of comprehension means the person is unable to mentally grasp the information. Misinterpretation implies that the person is able to mentally grasp the information but interprets it incorrectly due to any of several factors like – the person is erroneously confident that he/she has comprehended the information,

the information is ambiguous (has multiple possible interpretations) or when the presenter and the recipient have divergent perspectives on the information (Cioch, 1991).

3.2 Why Measure Comprehension?

There are many different visualization tools/techniques being implemented in the plethora of visualization systems available today. However, the widespread proliferation of visualization tools/techniques also highlights the need for their empirical evaluation. We are still lacking a measurement framework, which could objectively tell us the benefits of one tool/technique over the other for a specific task. “Once the visualisation has been designed and built it must be evaluated to see if it is capable of supporting the user in their tasks and meets all of the desired usability criteria” (Wilkins, 2003). The success of any visualization technique depends on the expressiveness and effectiveness of underlying graphical language in exploiting the capabilities of the output medium and the human visual system (Mackinlay, 1986). No matter how efficient a visualization technique may be, or how well motivated from theory, if it does not convey information effectively, it is of little use (Kosara et al., 2003). So, to quantify effectiveness i.e. to determine the extent to which a visualization system proves useful in practice, we need to measure it. Any visualization system is effective only if it is serving its main objective i.e. facilitate understanding of the underlying pattern in the data. Comprehension is crucial for overall effectiveness of any visualization system. This is the primary motivation for us to consider comprehension measurement.

On examining the literature, we found that the software professionals working on software visualization have also expressed the need for some benchmarks to evaluate the effectiveness of visualization systems. Rushmeier et al. (1995) state:

“There is also a general agreement that there are characteristics of visualization systems that make them very useful for some problems, and characteristics that make them essentially unusable for other problems. Unfortunately, very little work has been done to rigorously define what a good visualization or visualization system is. Currently, we have no measures to guide users in generating reliable, accurate and effective visualizations. Developers of visualization systems have no community-accepted standards and benchmarks to use in designing and validating their products. Purchasers of visualization software have no guidelines for comparing products.”

Therefore, in our research we are aiming to provide a measurement framework where we could express the comprehension of any visualization system by the intended user in quantitative terms. To assist us in the formulation of a measurement framework for comprehension assessment in visualization systems, we further studied the current state of measurement in software engineering, which we believe is the most closely related and also more advanced in this aspect.

3.3 Representational Theory of Measurement

Measurement has become a necessary tool to provide an objective vision on the quality of our daily activities. Thus, nowadays measurement is an integral part of any human activity whether it is social, economic, industrial, academic, environmental, medical, etc (Khelifi et al., 2004). In general, measurement is the process by which numbers or symbols are assigned to objects either real or abstract, that we observe in our intellectual environment. An example of a real object is a person (human being) and an example of an abstract object is an algorithm. Each instance of these objects has certain properties or attributes. However, the process of identifying the attributes of an abstract

object, like comprehension support provided by a visualization system, is not nearly so simple. Most of us have little or no training in the determination of the properties (attributes) of abstract objects. Thus, it is very difficult for us to measure these attributes.

Roberts in his book (Roberts, 1979) suggests that measurement has something to do with assigning numbers that correspond to or represent or preserve certain observed relations. A formal definition of measurement given by Fenton and Pfleeger (1997) is that it is a mapping from empirical world to the formal, relational world i.e. it is a process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to characterize the attributes by clearly defined rules. In the mapping, the real world is the domain and the mathematical world is the range. The mapping is called 'representation' or 'homomorphism' and it must preserve intuitive and empirical observations about the attributes and entities in the real world. The property of the entities that determines the mapping according to the prescribed rule is called a magnitude, the measurable attribute. The number assigned to a particular object by the mapping rule is called its measure, the amount or degree of its magnitude. The mapping rule will define both the magnitude and the measure.

The term metrics has been used widely to describe the act of measurement, and to imply the qualitative or quantitative performance indication. Its purpose is to accurately quantify an aspect of an existing or proposed system. Originally, a metric is defined as a criterion to determine the difference or distance between two entities, like the distance of two locations or the distance of a query and a document in information retrieval systems (Zuse, 1998). The Institute of Electrical and Electronics Engineers (IEEE) standard for Software Quality Metrics Methodology define the term 'software quality metric' as – “a

function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which the software possesses a given attribute that affects its quality” (IEEE, 1998). According to ISO/IEC 15939 (2007), measure is to make a set of operations having the object of determining a value of quantitative or categorical representation of one or more attributes. The term “metric” should be no longer used as synonymous of “measure” according to this standard.

The rest of this section is further divided into four parts – first part describes measurement models based on measures-based evaluation (an approach similar to what we will be using in our research), second part lists the measurement scales that can be used during measurement, third part explains the data types and methods of data collection, and the fourth part presents some of the evaluation strategies that could be applied for visualization systems.

3.3.1 Measurement in Software Engineering

Rombach (1991) states that in order for measurement to be successful, effective ‘top-down’ strategies that derive metrics from goals and interpret measurement data in the context of goals, are needed. We studied relevant models/standards in software engineering that are grounded in the theory of metrics-based evaluation. These models/standards are used by software engineers and usability specialists to measure the quality of software products. A brief explanation for each of them is as under.

□ McCall’s Model

McCall et al. (1977) proposed one of the earliest quality models. This model is also called GE (General Electric) model or FCM (Factor, Criteria and Metric) model. This model describes quality as being made up of a hierarchical relationship among the quality

factors, quality criteria, and quality metrics. The term “quality factor” is a key characteristic of the software product. “Quality criterion” is an attribute of the quality factor that defines the product. “Quality metric” denotes a measure that can be used to quantify the criterion. McCall et al. described a systematic approach to quantify quality as:

- Determine all of the factors that would have an effect on the software quality.
- Identify the criteria for judging each factor.
- Define metrics for each of the criteria and establish a normalization function that defines the relationship between the metrics and all the criteria pertaining to each factor.
- Evaluate the metrics.
- Correlate the metrics to a set of guidelines that every software development team could follow.
- Develop recommendations for the collection of metrics.

As depicted in Figure 3.2, McCall et al. identified 11 quality factors, 25 criteria and 41 metrics to measure these criteria. These metrics involved questions dealing with the degree of compliance to the criteria and had either a “yes” or a “no” for an answer. That is why the metrics results are highly subjective and it is generally difficult to interpret them.

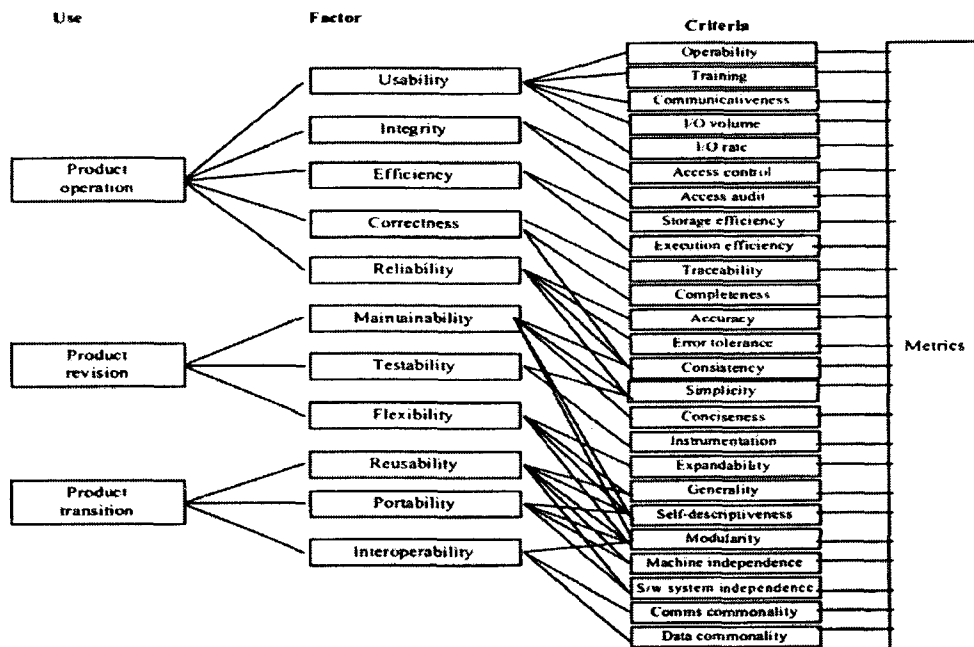


Figure 3.2: McCall' Software Engineering Quality Model

□ **Boehm Model**

This model was developed in 1978 by a team of researchers, lead by Barry W. Boehm (Boehm et al., 1978). Like the McCall model, this model also focuses on the final product. Both McCall and Boehm models assume that the quality attributes are at too high-level to be meaningfully measured, and therefore further decomposition is needed. The quality characteristics at a lower level are called quality criteria. In a third level of decomposition, the quality criteria are associated with a set of directly measurable attributes called quality metrics. Boehm's model of software quality is depicted in Figure 3.3.

It incorporates 19 quality factors encompassing product utility, product maintainability, and product portability. The criteria in McCall and Boehm models are not independent, and they interact with each other in a conflicting manner.

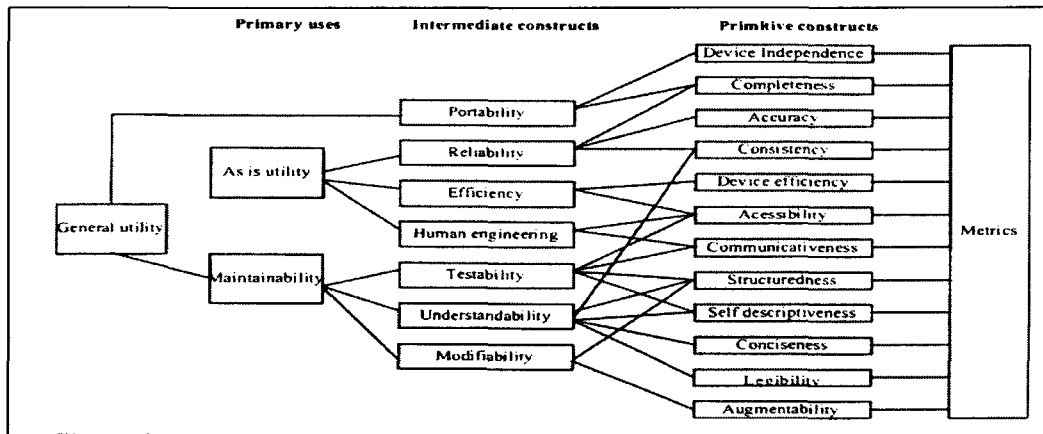


Figure 3.3: Boehm's Model

□ The GQM Paradigm

The Goal-Question-Metric paradigm was proposed by Victor Basili, as a means of measuring software in a purposeful way (Basili and Rombach, 1987). The main idea behind the GQM is that measurement should be goal-oriented and based on context characterization.

Goals are refined in an operational, tractable way into a set of quantifiable questions. Questions in turn imply a specific set of metrics and data for collection. GQM defines measurement at three levels (Figure 3.4):

1. Conceptual level (Goals): Goals are defined for an object based on specific needs, from various points of view and relative to a particular environment.
2. Operational level (Questions): A set of questions is defined for the model of the object that characterizes and assesses a specific goal.

3. Quantitative level (Metrics): A set of metrics based on the model of the object under study is defined for each question in order to answer it in a measurable manner.

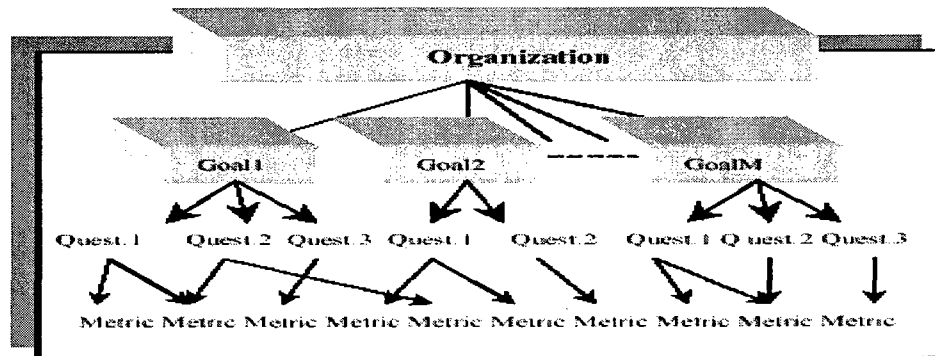


Figure 3.4: GQM Framework for An Organization

□ ISO 9126 Standard

For many years, there was a desperate need for a unique, unambiguous and usable software quality model. In 1991, an international standard was proposed for software quality measurement i.e. ISO 9126: "Software Product Evaluation: Quality Characteristics and Guidelines for their Use" (ISO 9126-1, 1991). This standard incorporates six quality characteristics – five of them (reliability, usability, efficiency, maintainability and portability) are similar to those in McCall’s model and sixth i.e. functionality (Are the required functions available in the software product?) is the new one. These quality characteristics can be further refined into sub-characteristics that can have measurable attributes. Revision of the model in 2000, introduced the concept of quality in use as the seventh software quality characteristic (c.f. Figure 3.5).

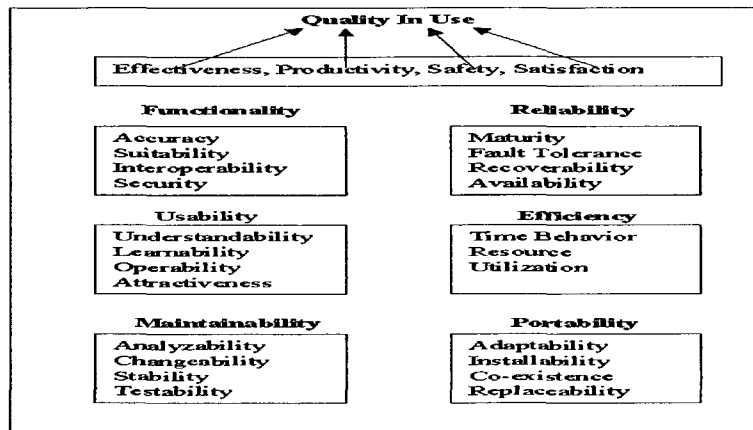


Figure 3.5: ISO 9126 –2000

Quality in use is the combined effect of the six software product quality characteristics and is determined in terms of the following four high-level quality attributes:

Effectiveness – The capability of the software product to enable users to achieve specified goals with accuracy and completeness in a specified context of use.

Productivity – The capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.

Safety – The capability of the software product to achieve acceptable levels of risk of harm to people, business, software, property or the environment in a specified context of use.

Satisfaction – The capability of the software product to satisfy users in a specified context of use.

□ **QUIM**

Quality In Use Integrated Map (QUIM) is a framework for specifying and measuring quality in use (Seffah et al., 2006). QUIM is also based on a hierarchical decomposition

like most other software engineering models. The difference is that it distinguishes five levels called – factors, criteria, metrics, data and artifacts for data collection purposes. The relationship between the elements of these layers is an N-M relationship. QUIM knowledge map is a repository of 10 factors, 27 criteria and more than 125 metrics for assessing quantitative as well as qualitative aspects of quality in use. Factors in this repository are applicable as generic characteristics of all software products, and the models are seen as specific subsets of this knowledge map, which have context specific characteristics. It further refines factors into measurable criteria and then maps the criteria into usability metrics. Empirical rules for understanding and interpreting metrics are also included in QUIM. This framework is also supported by a tool called QUIM editor. It allows the software developers to establish usability goals and create usability requirement specification. Software developers can evaluate the usability of their software products based on the specification.

3.3.2 Measurement Scales

When measurement is viewed as the mapping from the empirical properties to numbers (Zuse, 1998), the empirical and numerical relations are usually called the scale of the measurement. Scales provide values and units for describing the attributes of entities. For example, number of colors used in visualization is ‘4’; legibility of the text in visualization is “average” etc. Each of these observations has been quantified (or labelled) with a value from a (presumably) well-defined scale. Generally, there are five types of scales used in measurement as explained below.

a) Nominal scale

A nominal scale provides a name or label as the value for an attribute of an entity. Here, each entity is placed in a particular class or category based on the value of some attribute. The empirical relation system consists of different classes; there is no ordering among the classes. The only comparisons that can be made between variable values are equality and inequality. There are no "less than" or "greater than" relations among classifying names, nor operations such as addition or subtraction. Nominal measures are often used to classify entities so that they can be sorted prior to counting the number of occurrences or aggregating measured values. Examples are: first language of a person (English, French, other), color of a person's hair (red, brown, black, blonde, other), numbers for football players (nominal values limited to one player per number), and identifying attributes such as part numbers, job codes, defect classes etc. Among the admissible statistical functions for this scale, one can refer to frequency or mode.

b) Ordinal scale

An ordinal scale permits measured results to be placed in ascending (or descending) order. The empirical relation system consists of classes that are ordered with respect to an attribute. Any mapping that preserves the ordering is acceptable, e.g. comparisons of greater and less can be made, in addition to equality and inequality. The numbers represent ranking only; so addition, subtraction, and other arithmetic operations have no meaning. In addition to frequency or mode, one can use median as another statistical operation.

c) Interval scale

An interval scale adds the concept of distance. This scale captures information about the size of intervals that separate the classes, i.e. this scale preserves differences between any two of the ordered classes in the range of the mapping. Addition and subtraction are acceptable on the interval scale, but not multiplication and division. Meaningful statistics are the comparisons of arithmetic means, standard deviation, Pearson correlation coefficient and all that apply to ordinal scale.

d) *Ratio scale*

It is a measurement mapping that preserves ordering, the size of intervals between entities, and ratio between entities. The measurement mapping must start at zero and increase at equal intervals. All arithmetic operations can be meaningfully applied to the classes in the range of the mapping. Geometric mean and coefficient of variation are one of the appropriate statistical analysis techniques that can be applied on the ratio scale.

e) *Absolute scale*

The measurement for an absolute scale is made simply by counting the number of elements in the entity set. The attribute always takes the form “number of occurrences of x in the entity”. There is only one possible measurement mapping, namely the actual count. All arithmetic analysis of the resulting count is meaningful. Meaningful statistics are all that apply to above scales.

3.3.3 Data Types and Data Collection

Any empirical investigation results in data. Data can be classified into two categories – quantitative data and qualitative data. Quantitative data is expressed in the form of numbers and is obtained by assigning a numerical value or a symbol to a property or

attribute of a software engineering entity (Fenton and Pfleeger, 1997). Qualitative data (i.e. information expressed in the form of words and pictures) plays an important role in addressing the human aspects (Seaman, 1999). The advantage of qualitative data is that it is more informative. However, being subjective in nature it is more difficult to analyze this kind of data.

Data collection plays a crucial role in order to obtain insight and knowledge about software products (Zelkowitz and Wallace, 1998). A variety of data collection methods exist to collect data like – observation, interviews and questionnaires (Robson, 1993). An observation provides an opportunity to document activities, behaviour, and physical aspects without having to depend upon people’s willingness and ability to respond to questions. Interview is a kind of conversation with a specific purpose, for example to get an opinion of a person on a particular topic. The interview is a flexible and adaptable way of finding information (Freimut et al., 2001). Interviews are often distinguished based upon the degree of structure or formality of the interview (Robson, 1993). The fully structured interview applies a predetermined set of questions. The semi-structured interview applies a set of questions that have been worked out in advance, but the interviewer is free to modify the order based upon the perception of what seems most appropriate in the context of the conversation. During an unstructured interview the interviewer has a general area of interest and concern, but lets the conversation develop within this area. Questionnaires are a popular means of collecting data that are often difficult to design. Questionnaires comprise of open or closed ended questions.

3.3.4 Evaluation Methods in Software Engineering

There are many evaluation methods that are applied in software engineering and are also possible for visualization systems, as described in this and the next section. Each evaluation method may find different types of problems in a visualization system, and has its own benefits and drawbacks.

1. Controlled User Studies

User studies offer a scientifically sound method to measure visualization's performance (Kosara et al., 2003), and are particularly useful for evaluating the strengths and weaknesses of different visualization techniques. User studies involve real users to obtain both qualitative and quantitative data that is then used for calculating subjective and objective metrics respectively. Quantitative data typically measures task performance (e.g. time to complete a specific task) or accuracy (e.g. number of mistakes) and it can also be collected from user ratings on questions like task difficulty or preference. Qualitative data may be obtained through questionnaires, interviews, or observation of subjects using the system. According to Walenstein (2002) formal user studies can be time-consuming, expensive, and difficult to design. A clear objective, controlled experiment setting, and strictly-limited simple tasks are essential for drawing clear conclusions. Although, they quickly highlight problems in interfaces (e.g., it is easy to see whether a user can find the button to perform a task), user studies do not always identify problems and benefits of visualization ideas (Tory and Möller, 2004).

2. Usability Inspections

There are some other evaluation methods recognized in HCI (human computer interaction) which include – ‘cognitive walk-throughs’ where an expert ‘walks through’ a specific task using a prototype system thinking carefully about potential problems that could occur at each step, and ‘heuristic evaluations’ where an expert evaluates an interface with respect to several predefined heuristics (Mack and Nielsen, 1995). For example, Blackwell et al. (2001) describe ‘cognitive dimensions’ which is a set of heuristics for evaluating cognitive aspects of a system, and Baldonado et al. (2000) designed a set of heuristics specific to multiple view visualizations.

These usability inspection methods avoid many of the problems with user studies and may be beneficial for evaluating visualizations. However, usability inspection methods are (for the most part) designed for user interface testing and they limit our ability to find unexpected errors as they exclude end users from the evaluation process (Tory and Möller, 2004). Though, expert reviews can provide quantitative results without many resources and can be conducted in a short time, expert reviews should not be overly used, as the results of an expert review are limited by the experts' experience or their own personal biases. Therefore, an expert review should only be viewed as an alternative supplement to formal user studies.

3. Case studies of the tools in realistic environment (Plaisant, 2004)

This is an uncommon type of evaluation method, where the advantage is that users work in their natural environment doing their real tasks, demonstrating feasibility and in-context usefulness. However, the disadvantage is that they are time consuming to conduct and results are not repeatable.

4. Guidelines and Checklists

User interface could be evaluated based on the general design guidelines (Shneiderman, 1998; Nielsen, 1994). The conformance of user interface elements to these guidelines or checklists could be verified. The frequent use of ‘visualization mantra’ (Overview first, zoom and filter, then details-on-demand) is evidence that many visualization practitioners find it very helpful to evaluate different design scenarios (Craft and Cairns, 2005). However, researchers such as Welie et al. (2000) have noted, guidelines are often difficult to select, interpret and apply; they may be too simplistic, and they may even contradict each other.

3.4 Measurement of Visualizations

There has been a substantial amount of work done on how people comprehend information from graphs and visualizations in general (e.g. Kosslyn, 1989; Pinker, 1990; Tan et al., 1990; Trafton et al., 2000). However, lack of measures and evaluation techniques that give precise indications on the goodness of any visualization system is still an open problem (Bertini and Santucci, 2004). An exhaustive literature survey was conducted to find the related works that have been done by researchers in the field of measuring visualization systems. Unfortunately, this field is rather immature and there has not been a lot of work done on it. We were able to find only few studies relevant to our proposed research, which are as under:

3.4.1 The Visual Display of Quantitative Information (Tufte, 1983)

Description: Tufte was the foremost researcher who presented some preliminary work in this area. He proposed some measures to estimate the quality of 2D representations of static data. His work suggested measures like:

- 'data-ink ratio'- which represents the proportion of a graphic's ink devoted to the non-redundant display of data information,
- 'the lie factor'- that is the ratio of the size of an effect as shown graphically to its size in the data, and
- 'the data density'- that takes into account the size of the graphic in relation to the amount of data displayed.

Moreover, Tufte has explored 3D in his recent works (Tufte, 90; Tufte, 96) and has applied an extended version of these metrics to a 3D environment.

Relevance: These measures have been proposed for paper-based representations, and are not directly applicable to interactive, computer-based visualizations.

3.4.2 Metrics for Effective Information Visualization (Brath, 1997)

Description: Starting with Tufte's proposal, Brath defined new metrics for static 3D images. He has proposed heuristic guidelines and metrics for 3D interactive representations of business data. He has identified a few metrics to assess the efficacy of static 3-D presentations, which are:

- 'number of data points', i.e. the number of discrete data values represented on the screen at an instant,
- 'data density', that resemble Tufte's approach aiming at measuring visual image complexity given by number of data points divided by number of pixels,
- 'number of simultaneous dimensions displayed', which seeks to give an estimate of complexity by measuring the number of data attributes that are displayed at the same time,

- ‘occlusion percentage’, to provide a measure of occluded elements in the visual space, and
- ‘percentage of identifiable points’, i.e. the number of visible data points identifiable in relationship with every other visible data point.

Relevance: The proposed metrics are objective and fairly easy to measure. On the negative side, they are for static pictures and thus have not been extended to interactive models (Miller et al., 1997).

3.4.3 ViCo: A Metric for the Complexity of Information Visualizations

(Gärtner et al., 2002)

Description: The authors introduced ViCo, a metric for assessing information visualization’ complexity. The proposed metric allows for the measurement of information visualization complexity with respect to tasks and users. ViCo is actually an algorithm that allows a quantitative comparison of the relative complexity of a set of visualizations for any given situation. The authors conceptualize the complexity of visualizations in terms of the operations or cognitive elements that are needed to accomplish the tasks by users. The proposed metric of complexity does not deliver a single number but describes a function with various variables (e.g., number of items to be compared).

The algorithmic steps of ViCo (Visualization and Complexity) to develop the metric of complexity for a chosen set of visualizations are:

1. Analyze the tasks to be accomplished by the use of a set of given visualizations and select those tasks to be taken as the basis of measurement.

2. Define minimal reading, writing, comparing, and calculating operations with respect to users' groups and variables of the data set to be visualized.
3. Develop the functions that describe the number of such operations needed to accomplish such a task.

Relevance: This study reveals the importance of context (expressed by users and tasks) in any visualization. However, it does not provide the evaluation criterion by itself and asks the evaluators to analyze the visualizations to seek the basis of measurement. The authors also assume that the visualizations under consideration include all the information necessary to complete various tasks like minimal reading, writing, comparing and calculating operations. However, similar visualizations may vary substantially in what tasks they allow users to work on.

3.5 Lessons Learned

From above discussions, we draw two main lessons that will further lead us to formulate our measurement framework. These are as follows –

- In our proposed research, we believe that comprehension is a measurable quality factor that can be measured by using the same 'top-down' hierarchical manner, as applied by other models/standards explained in section 3.3.1 to identify useful measures. During the masters' work (Padda, 2003), we have worked on a hierarchical decomposition of quality in use factors into measurable criteria and metrics. We will use the knowledge gained from the masters' work to define criteria and measures to assess comprehension.
- We believe that using controlled experiment approach, as discussed in section 3.3.4, to derive the measurement results is a feasible and accurate evaluation strategy. The

hypothesis and the variables of the experiment are defined more clearly in the later chapters of this thesis.

Chapter 4. Eliciting Criteria for Comprehension

Measurement

"Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away." – Antoine de Saint-Exupéry (1900 - 1944)

Overview

In this chapter, we explore the means to achieve comprehension measurement and propose comprehension criteria that have resulted from the in-depth studies of earlier work (presented in chapter 2) addressing psychological, cognitive and visual communication aspects of a visualization system. Towards our main objective to measure comprehension support of visualization systems, we further decompose these high-level factors called 'aspects' into measurable criteria. The primary basis for these comprehensibility criteria is previous work by two renowned researchers – *Norman's Theory of Human Action Cycle* which describes how humans tend to interact with the outside world and *Effective Visual Communication* by Aaron Marcus which describe the principles that should be followed so that graphical user interfaces become effective media for communication with users. These principles aid in deriving an initial repository of comprehension criteria. Further, borrowing ideas from the area of non-functional requirements in systems engineering, we also present a verification scheme consisting of completeness, consistency, non-ambiguity, correctness and testability, enabling further refinement of these criteria. Experts having specialization in this field of comprehension have also been consulted in order to verify the proposed criteria.

4.1 Evaluation Foundation

To determine the criteria for each of the aspects (excluding 'Information Structure' aspect), we shall use well-defined principles from cognitive psychology and visual communication. These principles are the result of pioneering work of two eminent researchers – Donald A. Norman and Aaron Marcus, who have investigated the issues of human perception and cognition, and visual effectiveness respectively. The following paragraphs give a detailed explanation of these principles as applied to the identified aspects of comprehension in order to seek measurable criteria.

- **Norman's Cognitive Principles from the Theory of Human Action Cycle**
(Norman, 1990)

Donald A. Norman is a famous cognitive psychologist, who describes the psychology behind 'good' and 'bad' designs, through case studies, and proposes various design principles for understandability and usability. According to him, for a design to be successful, the system image should reflect a clear and conceptual model of the designer's view to the intended users as shown in Figure 4.1. In this figure, the design model is the designer's initial conceptual model of the system. The system image results from the physical structure that has been built using the available hardware and software resources. The user's model is the mental model developed by the user through interaction with the system. All communication between the design model and user's model takes place through the system image. If the system image does not make the design model clear and consistent, then the user will end up with an incorrect mental model.

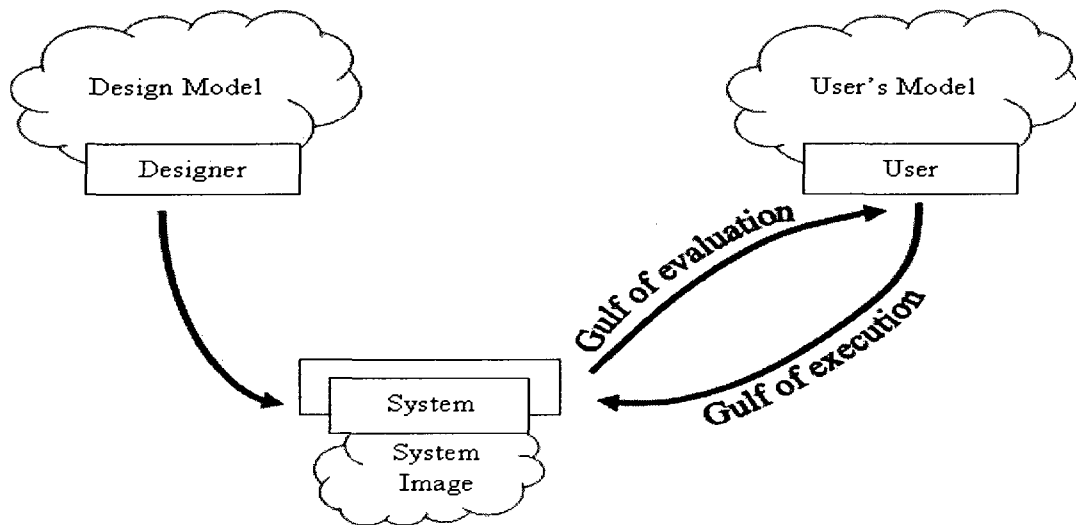


Figure 4.1: The Conceptual Models and The Gulfs

Furthermore, to form a mental model of the system, there are two gulfs that are encountered by a user while interacting with any system as shown in Figure 4.1. These are explained as below:

- ‘Gulf of execution’

It is the difference between the intentions of the user and the allowable actions of the system. One measure of this gulf is how well the system allows the person to do the intended actions directly, without requiring extra effort i.e. if the actions provided by the system match those intended by the person?

- ‘Gulf of evaluation’

It reflects the amount of effort that a person must spend to interpret the physical state of the system, and to determine how well the expectations and intentions have been met. This gulf is small when the system provides information about its state in a form that is easy to perceive, interpret, and matches the way the person thinks of the system.

To deal with these two gulfs, Norman proposes a set of design principles, including the principles of 'Naturalness of interaction or Mapping' and 'Affordances' as follows:

Naturalness of interaction or Mapping –

Natural Mapping is a term denoting the extent to which the relationship between two things, e.g. between the controls on screen and the actions, are apparent to the user. Natural mappings take advantage of physical analogy and cultural standards to guide immediate understanding. Natural mappings entail the least amount of efforts from the users' side in selecting the next action to interact.

Affordances –

Affordances are aspects of an object, which suggest how an object should be used, i.e. a visual clue to its function and usage. It means the perceived and actual fundamental properties of the object should determine how the object works. Affordances are essential for understanding the potential for interaction and manipulation in an environment. Well-designed objects are easy to interpret and understand, as they contain visible clues to their operation. Poorly designed objects can be difficult and frustrating to use, as they provide no clues or sometimes false clues. Poor design traps the user and thwarts the normal process of interpretation and understanding.

▪ **Visual Communication Principles (Marcus, 1995)**

Aaron Marcus, a renowned specialist in graphics design has proposed three basic principles to gauge the effectiveness of visual communication. According to him, an information-oriented, systematic graphic design helps the user to understand and process complex visual representations correctly. The design principles proposed by Marcus are

grounded on the pioneering work of Dondis (1973), who has proposed a number of principles for visual literacy. The key principles for effective visual communication proposed by Marcus are – ‘principle of organization’, ‘principle of economization’ and ‘principle of communication’ described as follows.

Principle of organization –

“Provide the user with a clear and consistent conceptual structure”. It signifies that the information presented to the user should be clear enough to perceive and understand easily. Consistency should be established internally within one user interface, externally across several, and in relation to real-world experience. The relationships among different parts of the information should be apparent, along with a clear primary and secondary focus for the user’s attention.

Principle of economization –

“Maximize the effectiveness of visual representation by using a minimal set of metaphors/cues”. It means one should include only the essential elements in order to make the visual representation more effective to the user. The simplicity and clarity of information should be focused by including only the essential elements, and by avoiding information ambiguity. The visuals should be made distinctive, and emphasized by distinguishing the important features.

Principle of communication –

“Match the presentation to the capabilities of the user”. It implies that the visual representation should also take into account the psycho-social needs, desires, education and other user-related characteristics. The visual design should ensure ergonomic design by establishing legibility, readability, and multiplicity of references (aliases).

We believe that these basic principles effectively cover the three aspects of comprehension (i.e. Visualization Interface, Perception, and Cognition) that we are concerned with, and therefore are appropriate to be used as the basis to seek appropriate criteria for the assessment of comprehension in visualization systems as discussed below.

4.2 Initial Repository of Comprehension Criteria

In my masters' work (Padda, 2003), comprehension was described as a potential factor to assess the 'quality in use' or the user perspective of the quality of software systems. It is considered as an important trait of visualization systems that affect their overall quality or value to the users. In general, comprehension is always prone to subjective interpretations unless it is quantified. In order to quantify the comprehensibility of a visualization system, one needs - to define criteria that the visualization system has to meet. Further for each of the criteria we must identify a set of measurable attributes, and finally measure them according to some specified procedure. Towards this endeavour, the first raw classification of criteria was proposed by one of masters' student in Concordia's human-centered software engineering research group as follows.

Joshi (2005) in her masters' thesis, proposed a set of comprehensibility criteria for modeling the comprehension gap between the user and visualization environment by combining two sets of principles from cognitive psychology and visual communication as explained in section 4.1. In a collaborative work with Joshi during two usability studies, an initial set of 19 comprehensibility criteria was identified as depicted in Figure 4.2.

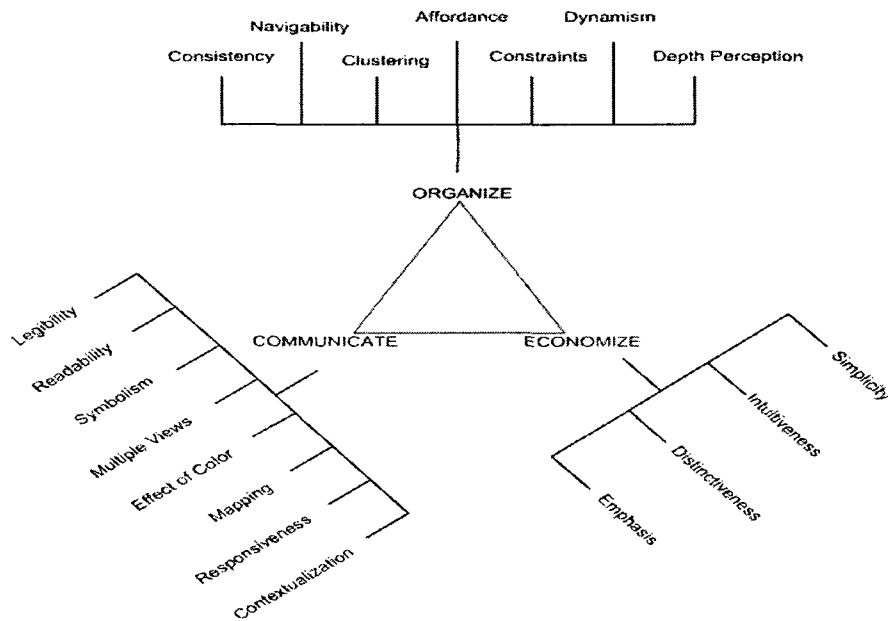


Figure 4.2: Assessment Criteria for Comprehensibility in Visualization Environments (Joshi, 2005)

This classification of comprehension criteria was based on the three basic visual communication principles proposed by Marcus (1995) i.e. organize, communicate and economize. The details of this classification scheme can be accessed in her thesis. We further refined this initial repository of comprehension criteria as follows.

4.3 Refining the Comprehension Criteria

In this section, we explain our approach towards the verification and refinement of the above proposed criteria for assessing user’s comprehension support provided by visualization systems. Comprehension criteria are the usability criteria of any visualization system and therefore come into the category of non-functional requirements of a visualization system that are important from the users’ point of view. With the lack of standardized verification techniques, non-functional requirements are rather considered as hard to quantitatively and objectively verify. Same is the case with our

criteria. However in software engineering literature some properties are provided in order to verify non-functional requirements. In our case, we have adapted these properties to verify the proposed criteria as shown in Table 4.1.

Table 4.1: Properties of Criteria

Property Name	Definition	How to achieve it?
Completeness	Determine if all the criteria needed to assess comprehension have been specified i.e. the criteria cover all aspects of users' comprehension. If any aspect is missing, it should be identified and described thoroughly.	This can only be done by a comprehensive analysis of related literature, which can later be refined by the experts' judgments.
Unambiguous	This requires us to check if the criteria are precise, clear and there is unambiguous interpretation. The meaning of each criterion has to be well-understood.	This property can be verified by describing each criterion at adequate level of detail so that readers can get a clear definition of the criterion. Moreover, we need to get experts' opinion on whether a criterion can be merged into other criterion/criteria in order to make it as a sub-criterion.
Consistent	It should be verified that no criterion conflicts with other criteria in the list. Related criteria should be kept together. The set of criteria should be internally consistent leading to a structured hierarchy of criteria.	We can verify for consistency by getting the opinion of experts in related fields.

Table 4.1 (continued)

Property Name	Definition	How to achieve it?
Correctness	It should be confirmed that the set of criteria is appropriate and not error prone in the sense that it does not contain any irrelevant criteria. This requires us to assess if the criteria are relevant to the problem in hand and all of them lead to measure some aspect of user comprehension in a visualization system.	In our research, we are verifying this property by analyzing the results of usability studies in different domains and verifying if we are able to extract certain features from visualization systems that could guide us to measure those criteria.
Testable	Can the criteria be tested?	In order to verify this requirement, we need to devise a set of questions independent of the domain and related to each criterion in order to measure them quantitatively or qualitatively.

In the following sections we provide detailed answers to all the questions related to each of the above listed properties.

4.4 Assessing Completeness by Studying Aspects of Comprehension

The original list of criteria proposed by Joshi (2005) (c.f. Figure 4.2) were categorized based on Marcus's visual communication principles (Marcus, 1995), and

were used by her in order to characterize the comprehension gap between visualization environment and the user. The formal verification of these criteria was not performed in her work. In our research, we want to quantify the comprehension support of any visualization system based on its' perceptual, cognitive and presentation qualities to the intended users. This is possible only by studying the relevant criteria for each aspect of comprehension as follows.

4.4.1 Categorizing the Comprehension Criteria into Aspects of Comprehension

This process of categorization is straightforward where we are taking into consideration the criteria proposed by Marcus and Norman for respective principles. In addition, we are also adding other criteria based on our literature survey of respective aspects of comprehension. This section illustrates our initial repository of criteria for each of the aspects, which were refined later by experts' opinion. The three aspects of comprehension i.e. Visualization Interface, Perception and Cognition are further categorized into measurable criteria as follows.

4.4.1.1 Visualization Interface

We need to study those characteristics or attributes of both - the interface and the visual representation, which contribute towards user comprehension. Marcus proposed that in order to understand the visual representation correctly, the visualization design has to follow the organization principle of visual communication. Kosslyn et al. (1992) also affirm this view as they say that pictures may be hard to fathom not only when they are too small or blurry, but also when the material in them is not organized in a way that we can comprehend easily. In case of software visualization systems, according to this

principle, visual representation should be organized in order for the users to get a clear and conceptual model of software structure. So, the organization and clarity of information matters, as it is not easy to distinguish when we have a mixture of many things. Organization is also related to overall layout of a visual representation that includes analyzing how easy it is to locate an information element in the display and to be attentive of the overall distribution of information elements in the representation (Luzzardi et al., 2004).

The main criteria introduced by Marcus (1995) that contribute to organization principle are – consistency, navigability, and spatial layout. A detailed explanation of each of these criteria suitably adapted in the context of visualization systems is given below.

1. Consistency

According to Jakob Nielsen (1997), “consistency is the key to usable interaction design”. A consistent visualization interface is the one that has an appealing look and feel and is easier for the user to operate because of the ease of remembrance and similarity of terminology on all screens. Consistency in labelling terms, actions’ output and structural representation of visuals do impact users’ comprehension.

Defining Consistency

Here are a few definitions of consistency that is considered as an important trait of all usable interfaces.

- ‘Agreement or harmony of parts or features to one another or a whole’ (Merriam-Webster, 2007)

- 'The degree of uniformity, standardization, and freedom from contradiction among the documents or parts of a system or component' (IEEE, 1990)
- 'Consistency means that similar user actions lead to similar results' (Wolf, 1989)
- 'Consistency refers to common actions, sequences, terms, units, layouts, colors, typography and more within an application program...' (Shneiderman, 1992)
- 'Attributes that bear on the visual uniformity of user interface' (Lin et al., 1997)

Types of Consistency

Grudin (1989) in his article "The Case against User Interface Consistency" states that there are three types of user interface consistency, which are:

- The internal consistency of a design with itself. User interface designers deal with internal design consistency by looking at consistency in physical and graphic layout, command naming and use, selection techniques, dialogue forms, etc.
- The external consistency of a design with other interface designs familiar to a user.
- An external analogical or metaphoric correspondence of a design to features in the world beyond the computer domain.

2. Navigability

With respect to a visualization system, the term navigability means the degree to which the user can steer through or manoeuvre within a visualization system i.e. the capability of the system to assist or direct the course of user's navigation. According to Marcus (1995) a visualization system should provide an initial focus for the user's attention, and then must direct further navigation throughout the visualization environment by providing attention to important, secondary, or peripheral items.

Defining Navigability

A few basic definitions of navigability can be given as:

- It is the degree to which a user can move around in the application.
- It is the ability to manoeuvre within a system.
- It is the ability of an interface to focus attention on the appropriate information and to lead one through the massive information.

Most visualization tools lack any form of navigational assistance, which would guide users through their information seeking process. For example, current navigational practices in most visualization systems are – clicking back and forward buttons, scanning the history list, selecting links through a combination of highlighted link text. Users are often guessing which link to follow next without any certainty of whether they are heading in the right direction. A problem which is often encountered in the use of large computer-based information systems is that of getting lost. For example, Elm and Woods (1985) define three categories of being lost in hypertext and hypermedia systems as:

- Not knowing where to go next
- Knowing where to go but not how to get there
- Not knowing a current location in relation to an overall context

According to Tory and Möller (2004) for a visualization system to have effective navigation, the following variables should always be visible.

- Cues should be present to help the user understand how to navigate through the display,
- Details at the current location,
- Details of the local neighbourhood, and
- Navigation history in terms of a list of previously explored display parameters.

3. Spatial Layout

The layout of any interactive visualization system consists of interaction objects (for example - list boxes, radio buttons, push buttons and so on) and interactive objects (for example - text, image, picture, video motion and so on) (Bodart and Vanderdonckt, 1994). Spatial organization or layout is concerned with - object location and spatial orientation, which tell us how easy it is to locate an information element in the display along with the context displaying the overall distribution of information elements (Luzzardi et al., 2004). In any visualization system, often the user wants to view a particular information object in detail while keeping its neighbourhood context visible on screen. Sometimes, locating an information element can be hard if the layout does not follow a logical organization and if some objects are occluded by others (Luzzardi et al., 2004). Luzzardi et al. (2004) also propose that degree of object occlusion and logical order are characteristics to be measured in visual representation concerning the location of objects. Spatial orientation is dependent on the display of reference context while showing details of one or more specific elements. Spatial organization can be measured by using qualitative measures to determine the ease in locating an object and the degree of awareness of the context (Luzzardi et al., 2004).

In addition to the principle of organization, principle of economization proposed by Marcus also impacts the visual interface effectiveness or user comprehension as it means inclusion of only the essential elements. The criteria proposed by Marcus (1995) which add to this principle are – simplicity, clarity, distinctiveness, and emphasis. These criteria are explained in the context of visualization systems as follows.

4. Simplicity

Simplicity is the quality of being uncomplicated or lack of impediment in accomplishing the desired goals. It means elimination of the extraneous and enhancement of user experience, while at the same time not sacrificing the quality of information. Anything in addition to the necessary detail distracts the visual message and confuses the users (Joshi, 2005). The visual design should display the most important controls, objects, and group related tasks together offering only a few choices at any time. It also depends on the visual designer's intention; the intention should be avoidance of confusion, even at the expense of beautification or attractiveness. The word 'simplicity' can be interpreted through three dimensions – functionality reduction, understandability and ease of use of application. The central idea behind simplicity is that users will feel more pleasure in their experience and have more positive reactions to a software system.

Defining Simplicity

A few basic definitions of simplicity are:

- Simplicity means lack of complexity or lack of impediment in accomplishing the user-defined goals
- It means to eliminate the extraneous and enhance the user experience, while at the same time not sacrificing the quantity of information.

Types of Simplicity

The design principles set by Cognetics Corporation (Kreitzberg, 1998) states that several types of simplicity contribute to a well-designed user interface, which are:

- *Visual simplicity* is achieved by showing only the most important controls and objects. Screen layout should follow good visual design practices. Use white space as a visual element to define perceptual areas.
- *Verbal simplicity* comes from the use of direct, active, positive language.
- *Task simplicity* is achieved when related tasks are grouped together, and only a few choices are offered at any one time.
- *Conceptual simplicity* is accomplished by using natural mappings and semantics, and by using progressive disclosure.

5. Clarity

The second factor for achieving screen economy is clarity, i.e. to design all components so that their meaning is not ambiguous (Marcus, 1995). Cioch (1991) also states that information ambiguity (has multiple possible interpretations) can cause misinterpretation and decrease the level of comprehension as well.

Types of Clarity

According to Dudycha (2003), clarity can be achieved in two ways as follows.

- *Conceptual clarity* depends on the visualization designer having a clear understanding of the phenomena being represented in the visual(s). Only with a clear understanding of the concepts involved in the problem at hand would it be possible to design a solution that not only showed the spatial distribution of conceptual entities but also revealed something of the underlying processes and spatial interrelationships among those entities. Conceptual clarity requires a clear statement of goals and understanding of the spatial patterns and processes to be represented on the visual(s). Conceptual clarity is rendered into the graphic design of visualization through careful

selection of important information, eliminating any unnecessary detail, and including a legend that identifies the intended meaning of all icons/symbols used in the visual(s).

- *Visual clarity* refers to the transformation of software features into graphic symbols on the visual. A well-conceived visual may still be poorly understood if the choice of icons/symbols used is not thoughtfully considered. Visual clarity can be improved by avoiding overlapping icons/text and lines, using a small number of related symbols or patterns, limiting the number of colors or fonts on screen and also selecting the icons/symbols according to their cultural meanings.

6. Distinctiveness

In order to achieve screen economy, it is also important to distinguish important properties of essential elements. Essential elements must stand out based on perceptual attributes like color, brightness, texture etc., and appear distinct. Distinctiveness can be achieved if there is less similarity among concepts. According to Schmidt (1991), distinctiveness enhances memory by increasing the saliency of the relevant information. Distinctiveness also promotes the use of visual techniques to direct the focus of the user to important objects or parts of the scene (Wickens, 1992).

Types of distinctiveness

Kurniawan (2000) has worked on the visual distinctiveness of icons, and proposes that icon's distinctiveness could be divided into two types as under.

- *Physical distinctiveness* is related to recognition of the objects the icon is comprised of, and it can be improved by –
 - a) maximizing the size of the objects in the icon,

- b) minimizing the spatial frequency of gratings (“a grating is any regularly spaced collection of essentially identical, parallel, elongated elements” (Wikipedia, 2007)), and
 - c) minimizing the use of color.
- *Perceptual distinctiveness* is related to understanding of what the objects in the icon represent. Perceptual distinctiveness can be improved by –
- a) maximizing the familiarity of the objects used in the icon, and
 - b) maximizing the clarity, uniqueness, and completeness of the objects in the icon to represent its referent.

7. **Emphasis**

The dictionary meaning of word ‘emphasis’ means – to accent the appearance, to underline, to put in bold, make something more significant or important (Wikipedia, 2007). Emphasis, in typography is defined as the visual enhancement of a part of information to make it noticeable. Emphasis refers to the visual process of accentuating important messages to the user in order to direct user attention to an important event or scene within a visualization artifact (Bugajska, 2005).

According to Marcus (1995), to achieve screen economy it is essential to make the most important elements salient, i.e. easily perceived. This can be done by de-emphasizing non-critical elements and minimizing clutter so that critical information is not hidden. Features in visualizations systems that are likely to catch attention are those that are brightly coloured, moving or changing, defined by sharp boundaries, or are highly saturated (Rheingans and Landreth, 1995). In a visualization system, we want to

observe if such an emphasis affects user's attention i.e. if they are able to observe the critical elements of visualization that are being emphasized (Joshi, 2005).

Figure 4.3 summarizes all the criteria and sub-criteria related to visualization interface aspect.

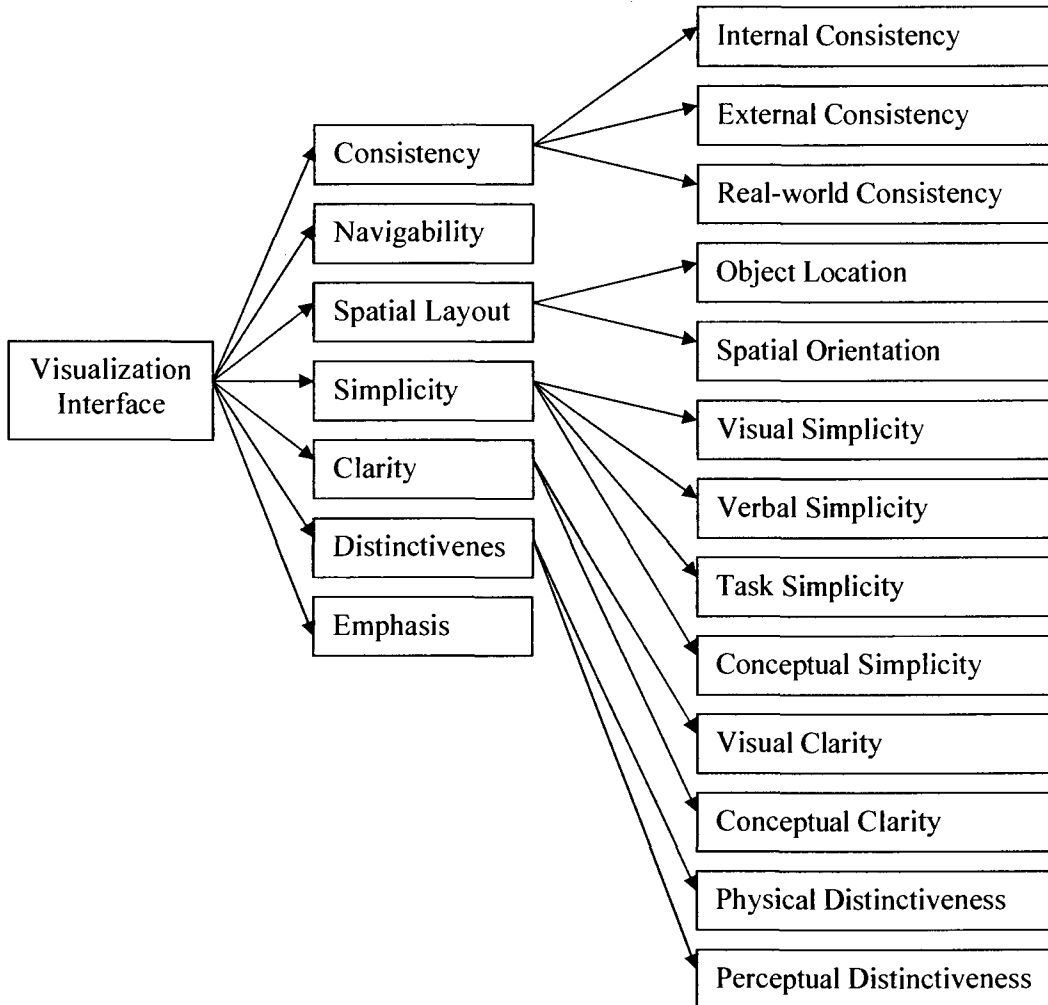


Figure 4.3: Comprehension Criteria for Visualization Interface Aspect

The second aspect 'perception' is categorized into criteria as follows.

4.4.1.2 Perception

Gleaning from literature on visual perception, we again propose a number of criteria that can be qualitatively assessed to determine the impact of visual perception on human comprehension. These are defined as under.

1. Affordance

James Gibson (1979) in his ecological approach to visual perception states that the environment is perceived by an individual as a set of *affordances*, i.e. ‘the actions a given environment affords to a given acting observer’. Thus, according to this theory, perception and action are tightly coupled. A user who perceives correctly will be able to perform correct operations or actions. This idea of affordances was later formulated by Norman as one of the cognitive principles that affect visual perception. Norman defines affordances as - aspects of an object which suggest how an object should be used; a visual clue to its functions and usage (Norman, 1990). It says that the perceived and actual fundamental properties of the thing are the ones that determine how the thing could possibly be used. According to Norman, perceived affordances are essential for understanding the potential for interaction and manipulation in the environment. Affordance provides strong clues to the operation of objects as users can figure out what to do by just looking at them (Norman, 1990). For example, a software package if it is being represented by a file folder metaphor then it clearly indicates to the user that it contains a collection of files. Similarly, having a plus sign in front of a node indicates that it can be expanded further. Affordance is supported by previous experience or learning of how to use the particular interface. If the visualization system has a familiar feel to it, the users can be surer of what they are doing. They feel safe interacting with the visuals,

knowing that something unexpected would not happen. Thus, they get the feeling of control. Affordances are the means of communicating the design model to the user, and designers can evaluate a system in terms of functions that they made clear or emphasized to the users through affordances (Mohnkern, 1997).

2. Symbolism/ Metaphors

All visualization systems make use of certain metaphors that act as a mapping between the visual components used in the realization of visualization and the underlying data set. The design of the metaphor can greatly influence the usability of the visualization (Knight and Munro, 2001) and hence the user understanding. Marcus (1994) states that metaphors are the figurative similarities of fundamental concepts, terms, and images by which and through which information is easily recognized, understood, and remembered. Visual symbols are used to describe ideas, and interaction semantics. The advantage of symbols/metaphors is that they address everyday experience and facilitate understanding of the content being portrayed through visuals. Good metaphors in user interfaces enable users to comprehend, use, and retain information more quickly, with greater ease, and with deeper satisfaction by effectively managing the users' expectation and comprehension (Marcus, 1998).

3. Dynamism/Animation

The human visual system is extremely sensitive to motion or kinematics. Animation is a particularly salient attribute of our peripheral vision capability (Pfitzner, 2003), and it is a suitable method to represent dynamism. Being a pre-attentive visual feature, animation is particularly suited to attract the user's attention and its detection happens at the early stages of visual perception (Ware, 2000). Motion has been extensively used in

psychology to extend the viewer's ability to perform basic exploration tasks. In visualization systems, animation can be employed to highlight information which is particularly important for the user to perceive quickly. Therefore, animation can be used in the fast, pre-attentive visualization of complex data (Healey et al., 1995; Healey et al., 1996) or for filtering and brushing techniques in visualization systems (Bartram and Ware, 2002). It allows moving patterns to pop out, and aids in the identification of a focal point in the visual by potentially alleviating visual interpretation complexity (Burd et al., 2002). In software visualizations for example, especially dynamic visualizations, use of animation is extremely important as programs are fundamentally dynamic and animations helps to illustrate how the program changes from state to state and how the software program evolves over time (Mukherjea and Stasko, 1993). According to (Nakakoji et al., 2001), in a visualization system a user interacts with animated visualizations in order to

- identify data points where the values change prominently,
- find a snapshot of a particular point of time, and
- acquire a feeling of immersion to more intuitively understand data.

4. Appearance

The dictionary definition of the word appearance means the visible aspect of a thing. We use this word to represent the user perception of the visual in terms of its visual attributes. According to Smolnik et al. (2003), the perception of an information-transmitting stimulus is a prerequisite for processing presented information. Therefore, we believe the appearance of visual objects is one of the more important criteria in the process of complete comprehension, as what appears on screen is what is perceived. We want to study those features of visualizations that make their appearance to be

comprehended readily. All visualizations are composed of certain basic visual attributes like color, line orientations, transparency, position, size etc. These basic visual attributes are utilized for performing more complex visual tasks like perception of shape, Gestalt, and depth (Ferweda, 1998). Shape perception is highly dependent on orientations (Wünsche, 2004) and is derived from luminance, motion, binocular disparity, color and texture (Davidoff, 1991). Perception of gestalt is influenced by proximity, similarity, closure, symmetry, common fate and continuity laws. Depth perception is achieved through both binocular vision using stereo goggles or head mounted displays, and visual clues like- size, brightness, textures, perspectives (Wünsche, 2004). As one of criteria, we want to observe if users are able to perceive the appearance of different objects represented using various visual attributes.

The criteria for the perceptual aspect of comprehension are depicted in Figure 4.4.

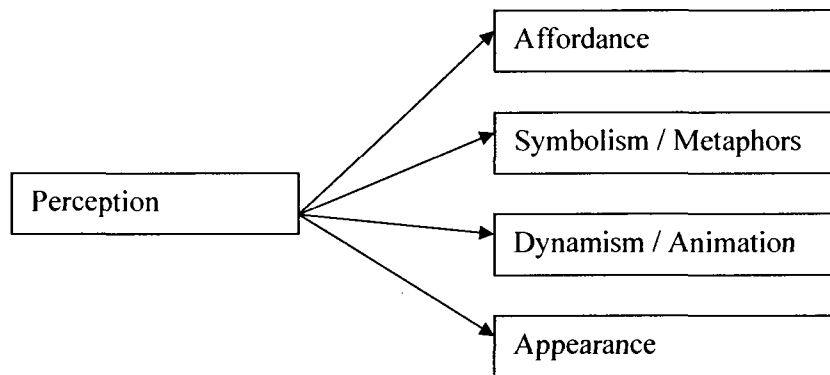


Figure 4.4: Comprehension Criteria for Perception Aspect

The final aspect to consider for comprehension is cognition and is further classified into measurable criteria as discussed below.

4.4.1.3 Cognition

The visualization tool/technique should provide an ergonomic design that matches the cognitive capabilities of the user. This is what is stated in the third principle of effective visual communication by Aaron Marcus (1995). To ensure ergonomics properties of any visualization system that affect human cognition, Marcus (1995) has proposed a set of criteria like – legibility, readability, multiple views, effects of color. These criteria are explained further.

1. Legibility

Adapting its definition originally from city planning, legibility for visualization systems can be defined as the ease with which people are able to learn the layout of visualization and then use this knowledge to perform wayfinding tasks (Ingram and Benford, 1996). According to the linguists, legibility determines whether an item can be read or deciphered, i.e. whether it is capable of being read (Haramundanis, 2001). It means the individual characters, symbols, and graphic elements should be easily noticeable and distinguishable. It has been noticed that character size and luminance contrast affect legibility of text on screen (Ayama et al., 2007). Inadequate contrast frequently occurs when the background and text color are similar. As a general rule, the darker, spectrally extreme colors such as red, blue, magenta, brown etc. make good backgrounds while the brighter, spectrum-centered, and de-saturated hues produce more legible text (Baecker et al., 1995). Moreover, the environment in which the visualization system is used also significantly affects the legibility of displayed visualizations. According to Baecker et al. (1995) dark screen backgrounds in brightly lighted rooms

may cause distracting reflections that can diminish screen legibility and in contrast, brightly lighted screens in dark rooms may be too glaring and hard to see.

2. Readability

The term readability means that the display is comprehensible, i.e. easy to identify and interpret, as well as inviting and attractive (Baecker et al., 1995). The concept of readability incorporates the interaction or engagement of a user with the system (Kane et al., 2006). The term applies both to the text and graphics. Readability of text is dependent on a number of factors like – its' syntactic difficulty, semantic difficulty, legibility and text organization (Chall, 1958). The readability of a graphic representation can be defined as the relative ease with which the user finds the information he/she is looking for (Ghoniem et al., 2004). According to Entin and Klare (1985) apart from readability of text, reader's comprehension of text is influenced by readers' levels of interest and their prior knowledge as well. On the graphics side, readability studies have been performed for 2D graph drawing (Purchase et al., 1996), where it has been seen that the readability is associated with (often conflicting) aesthetic criteria such as – minimization of edge crossings and area of the graph, and the maximization of symmetries.

3. Multiple views

Multiple views provide multiple perspectives of the visual representation in order to make it easier to understand. A multiple view system uses two or more distinct views to support the investigation of a single conceptual entity (Baldonado et al., 2000). By looking at multiple views of an object simultaneously, users are helped to get a clearer picture of the structure of the object. Same concept or object can be shown at different levels of abstraction in order to comprehend it at various levels of detail, like representing

the software architecture as the component hierarchy or code hierarchy abstractions. Multiple views offer a variety of benefits like – discovery of unforeseen relationships, improved user performance and so on (North and Shneiderman, 1997). Multiple views significantly impact cognitive overhead including time and effort to learn a system, load on user’s working memory, effort required for comparison tasks, and effort required for context switching (Baldonado et al., 2000). On the other hand multiple views minimize some of the cognitive overhead engendered by a single complex view of data (Baldonado et al., 2000). The facility to see multiple views at once provides cognitive support (Walenstein, 2003) as it reduces the memory load on the user.

Many renowned cognitive scientists (e.g: Card et al., 1999; Norman, 1993) have studied how visual representations aid in cognition and the principle of naturalness is proposed by all for developing effective visual representations.

4. Naturalness of interaction / Mapping

Defining Mapping

- Dictionary definition of the word mapping states it as a correspondence by which each element of a given set has associated with it another element(s) of a second set.
- In the context of visualization system, mapping means the natural relationship between user’s actions and their effect on visual representation.

According to Norman (1990), “Mapping is a technical term that relates actions and results... Natural mapping takes advantage of the physical analogies and cultural standards that lead to immediate understanding... and natural mappings require the least amount of effort from user’ side in deciding the next action to interact.” The author

argues that natural mappings be suitably exploited so that user can determine the relationships:

- Between intentions and possible actions,
- Between actions and their effects on the system,
- Between actual system state and what is perceivable by sight, sound, or feel, and
- Between the perceived system state and the needs, intentions, and expectations of the user.

According to Norman (1993), experiential cognition is most effective when the properties of the visual representation most closely match the information being represented. In order to achieve natural mappings the metaphor used in the interface should be most appropriate (i.e. natural) for the application domain. The terminology used in the interface should also be based on the application domain's terminology.

The criteria for the cognition aspect of comprehension are presented diagrammatically in Figure 4.5.

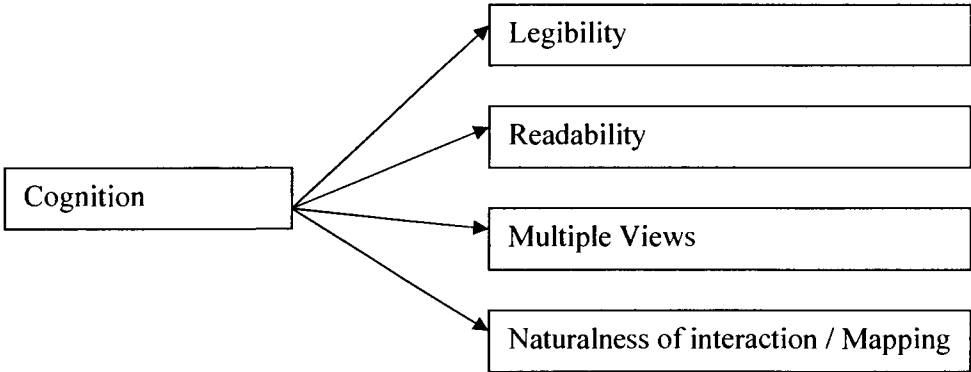


Figure 4.5: Comprehension Criteria for Cognition Aspect

4.4.2 First Iteration – Aftermaths

Putting together all the criteria for comprehension assessment discussed in the previous section, Figure 4.6 shows all of them along three dimensions i.e. Visualization Interface, Perception and Cognition.

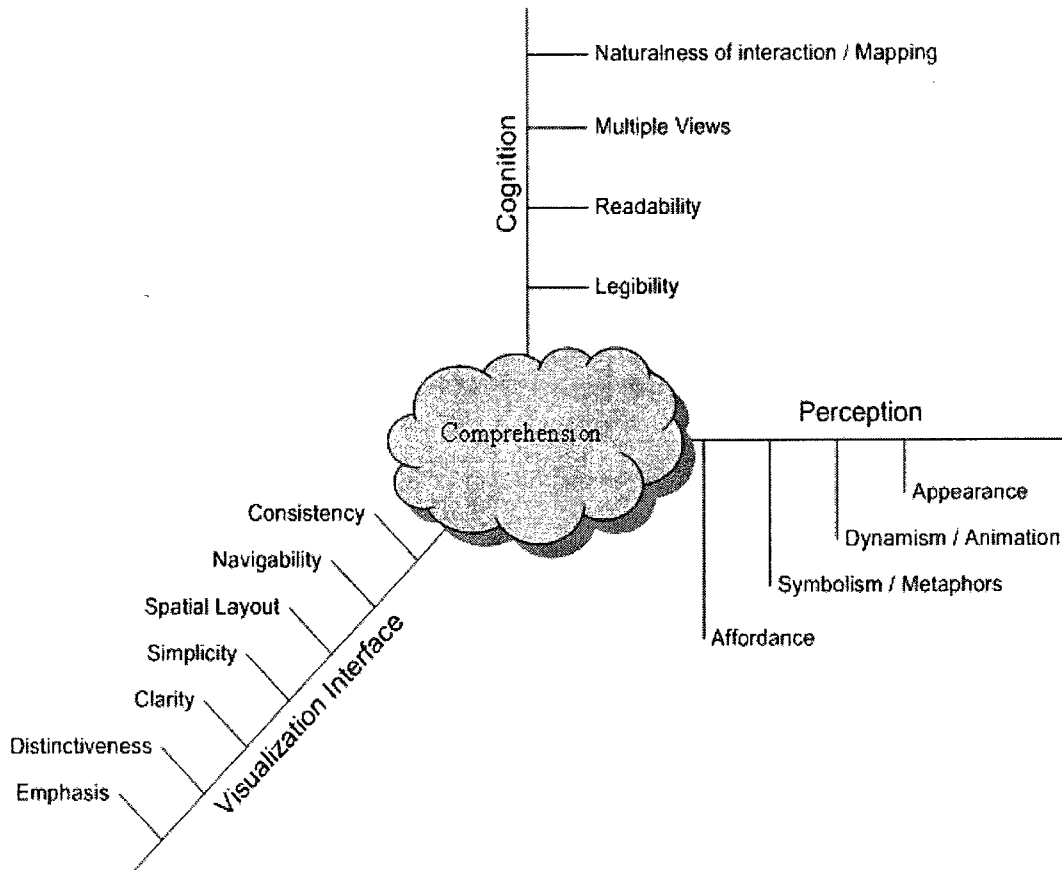


Figure 4.6: Criteria for Comprehension Assessment in Visualization Systems

When comparing Figure 4.2 and Figure 4.6, one can see that criteria *Clustering*, *Constraints*, *Depth Perception*, *Effects of Color*, *Contextualization*, and *Responsiveness* have apparently been excluded in this version of comprehension criteria. A detailed explanation for these changes is as follows:

- In our work, we have merged criteria *Clustering* and *Depth Perception* together and named it as criterion *Appearance*. The *Appearance* criterion is about all those visual properties that are designed in the system with the intention to show some representative features of the dataset like - clustering of similar objects or depth factor of an object. The visualization(s) uses some of the basic visual attributes to depict these features of the dataset.
- The criterion *Constraints* in Joshi (2005)'s thesis is actually a part of *Affordance* criteria in this refined version. The *Constraints* is defined as the forces, conventions that confine the set of possible actions. This definition by itself relates to *Affordance*, which is about the possible uses, actions and functions of an object. According to Norman (1990), "*Affordances* suggest the range of possibilities; *Constraints* limit the number of alternatives". Therefore, we believe that the criteria *Constraints* can be merged in criteria *Affordance* to describe the possible interaction mechanisms in visualization systems.
- *Effects of Color* by Joshi (2005) was described as one of the criteria for communication principle. We believe that color is a basic visual attribute that affects many of other criteria like - *Emphasis*, *Distinctiveness*, *Legibility*, *Readability*, *Appearance* etc., and therefore cannot be described at the level of other criteria. Hence, the criterion *Effects of Color* is used as a measure to assess these criteria in the refined version.
- *Contextualization* in Joshi (2005)'s work is expressed in the form of "focus + context" or "overview + detail" interaction mechanisms used in various visualization techniques. She said that in a visualization environment, often, the user wants to view

a particular part of complex visual representation in detail while keeping its context visible. This criterion deals with more about the spatial context and the navigation mechanisms that are provided in the visualization system, and therefore this is already included in our revised version under the *Spatial Layout* and *Navigability* criteria.

- The criteria *Responsiveness* is the quality of being responsive to user actions. We believe that this criterion is an effective parameter to determine the utility of a visualization system. A visualization system if it is not properly responding to a user action is likely to be less useful or usable, and users will not probably use it regardless of the comprehensibility of displayed visualization(s). Therefore, we were not sure of its inclusion in this revised version and for the purpose of second iteration we do not consider it further. We needed to seek expert opinion on this criterion and where it fits in overall distribution of criteria in one of three basic dimensions (i.e. Visualization Interface, Perception and Cognition in Figure 4.6). The accomplishment of this task of expert judgment is described in the forthcoming section where this criterion is further verified.

These criteria depicted in Figure 4.6 are further verified for their unambiguous, consistency properties by seeking the experts' opinion in related fields as follows.

4.5 Confirming Un-ambiguity, Consistency – Experts' Opinion

In the second iteration for verification, we met two experts having expertise in relevant fields. The first one was consulted to seek his viewpoints on the criteria for 'Visualization Interface', and second expert was asked to judge the criteria for both

'Perception' and 'Cognition' aspects respectively. Our approach to experts' selection was based on our three basic aspects of comprehension i.e. Visualization Interface, Perception and Cognition. One of the experts was specialist in User Interface Design Fundamentals and human factors related in HCI field, other was a professional in cognitive and psychology side of the users. A detailed explanation of characteristics of our experts in significant domains is as follows.

Visualization Interface expert – An associate professor of computer science at a university level. His research interests are in human-centered software engineering. He is a member of many professional advisory committees. He has more than 10 years of experience in the field of human factors, user interface design and empirical studies. He is also the writer of numerous articles in scientific proceedings, journals and book chapters.

Cognition and Perception expert – An associate professor of psychology at a university level. She was an associate director of the center for usability in design and assessment. She has more than 8 years of expertise in areas of attention, perception, cognition and human factors. She has over 50 publications in areas relating to human performance, human factors and human-computer interaction.

After this second step of verification, the judgment of the experts was used to further refine our proposed set of criteria. The summary of findings based on expert consultations is given below.

4.5.1 Summary of the Experts' Findings

Our experts have suggested the following changes in our criteria.

▪ **Changes in Visualization Interface criteria**

- Our expert on Visualization Interface aspect recommended removing *Consistency* and *Spatial Layout* criteria, as according to him both the criteria are feeding other criteria on Visualization Interface along with Perception and Cognition dimension. He also pointed that both of these criteria are properties of the visualization interface, and are not the quality attributes to measure comprehension from users perspectives. He states that for a visualization system to be *simple, legible, and easily perceptible* in terms of *appearance*, it needs to be *consistent*. He further added that *consistency* contributes to *navigability* by giving an example of city map. He said that if the signs displayed on the street boards are not consistent then a person will encounter great difficulties in reaching his/her destination. He also expressed his opinion to remove *spatial layout* criterion, as this is a sub-criteria which is needed in order to have a good *navigational* mechanism and to achieve task *simplicity* in a visualization system.
- The expert also suggested to rename *Navigability* as *Reachability*, as navigability is mostly defined as a property of interaction and can be calculated using some simple formulas. For example, in graph theory it can be measured by distance between two nodes. The term *Reachability* is more appropriate to express the navigation mechanism of a visualization system. Reachability refers to the possibility of navigation through the observable system states i.e. whether the user can navigate from any given state to any other state.
- He also proposed to rename the term 'Visualization Interface' as *Presentation* in order to signify the presentation qualities of both an interface and visualization(s) that affect the users' comprehension.

▪ **Changes in Perception and Cognition criteria**

- The expert on cognition and perception suggested that we merge *Affordances* and *Symbolism/Metaphors* criteria together and name them as *Affordances* only. This is because metaphors are a means to convey certain affordances. Designers create appropriate visual affordances via metaphors. Affordances communicate the design model to the users through the use of metaphors that the users are already familiar with. This suggestion is also supported by another researcher Ken Mohnkern (1997), who says in a bulletin for Special Interest Group on Computer-Human Interaction, “We can readily see the relationship between metaphor and affordance. When a metaphor is applied to a system, it gives the system a particular set of affordances. Metaphor is a container for a particular set of affordances.... When we create an interface metaphor, we are, in essence, dumping the contents of the metaphor (its affordance set) onto the computer system. Some of those affordances fit nicely onto the system's feature set (else that metaphor would not have been chosen), others do not have a corresponding feature in the system, and some of the system's features are left affordance-less, invisible.”
- The expert also suggested that we combine *Readability* and *Legibility* criteria into one criterion called *Legibility*. According to the expert, both terms are used interchangeably; however *Legibility* appears to be at a level higher than *Readability*. This viewpoint is also supported by Aernout de Beaufort Wijnholds (1996), who states that –
“The difference between the legibility and readability may best be expressed in terms of their relationship. When a text is of low legibility, its readability is also low. When

a text is not very readable, on the other hand, it is still possible that it is highly legible. Consider an instruction manual, for example, of which the characters are hardly identifiable; the print is so small and the characters have such indistinct shapes, that readers can hardly distinguish between the 'i' and the 'l' or the 'h' and the 'b'. In such a case, the text is of low legibility. As a result, the text is not very readable either. Even if a clear distinction can be made between separate words and if it is clear which part of the text corresponds to which drawn illustration, this will be of little use to the reader. If the instruction manual was reprinted in a more legible way, the same conditions of easy word distinction and correspondence between text and illustrations would make a more readable text. It is also possible, however, that the text has become highly legible, but that the illustrations are not numbered and are referred to in the text on a different page. In this case, readability would be low.”

- The expert also suggested labelling *Dynamism/Animation* criterion as *Dynamism* only, because animation or motion is one of the techniques to show the dynamics of the data set. According to Pfitzner et al. (2003) dynamic variations can be shown by – animating or moving objects, changing the sizes of objects displayed on screen, and changing the brightness or color etc.
- The criterion *Naturalness of Interaction/Mapping* was suggested to be labelled as *Mapping* only because naturalness of interaction is an internal property of a chosen mapping.
- The criterion *Multiple Views* was renamed as *Perspective-ness* in order to express it at a higher level where multiple views could be applied to show different perspectives of the visual entities.

4.5.2 Second Iteration- Aftermaths

Based on our experts review, our second revision of comprehension criteria is illustrated in Figure 4.7.

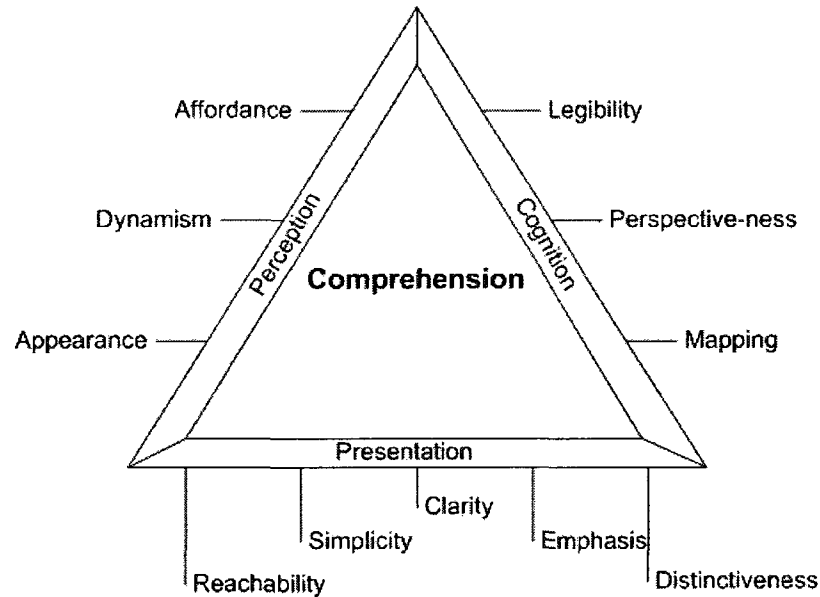


Figure 4.7: Refined Comprehension Criteria

These criteria impact one another as the three aspects of presentation, perception and cognition are inseparable when it comes to comprehension, as to understand anything we need to perceive its presentation and then we need to use our cognitive capabilities to fully comprehend it.

Chapter 5. Case Studies

“The man of science has learned to believe in justification, not by faith, but by verification.” – Thomas Huxley (1825-1895)

Overview

For the further verification of the criteria formulated in the previous chapter, two different visualization systems were evaluated as preliminary case studies. The first system was a three-dimensional bioinformatics visualization tool, and the second was an immersive art visualization environment. Two usability studies were conducted in Concordia’s human-centered software engineering lab with these systems, where participants were invited to use these visualization systems and required observations were made.

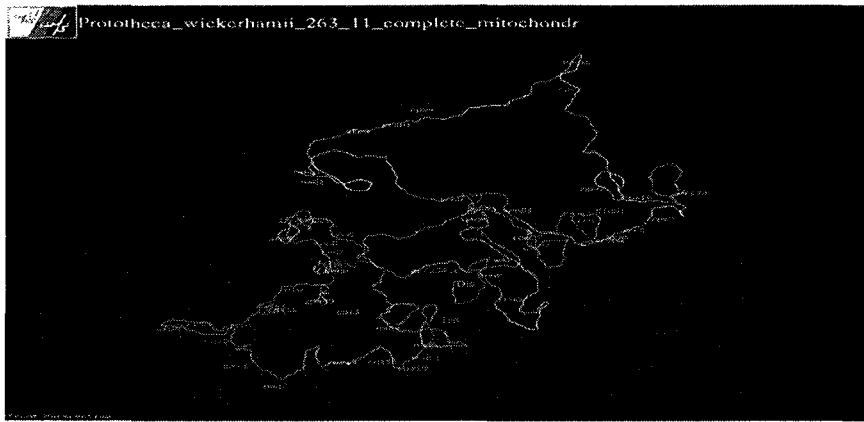
In this chapter, we explain in detail the conduct of these studies along with the analysis of their results, and finally we present measures and a measurement scheme to analyze the results of our comprehension criteria.

5.1 Verifying Correctness: Case Study 1

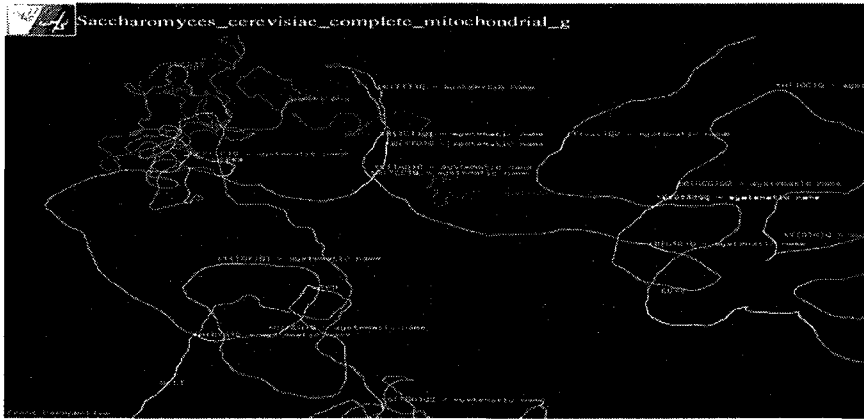
In order to verify correctness of criteria, we analyzed the applicability of our refined list of criteria in the usability results obtained from a collaborative controlled experiment conducted with Joshi (2005) for a bioinformatics visualization tool. This was done in order to determine the appropriateness of criteria in judging the users' comprehension of a visualization system. The details of this study can be accessed in her thesis (Joshi, 2005). The research results presented in this chapter are the contributions of the author. A summary of the analysis conducted for the purpose of verification of criteria is as follows.

5.1.1 ADN-Viewer

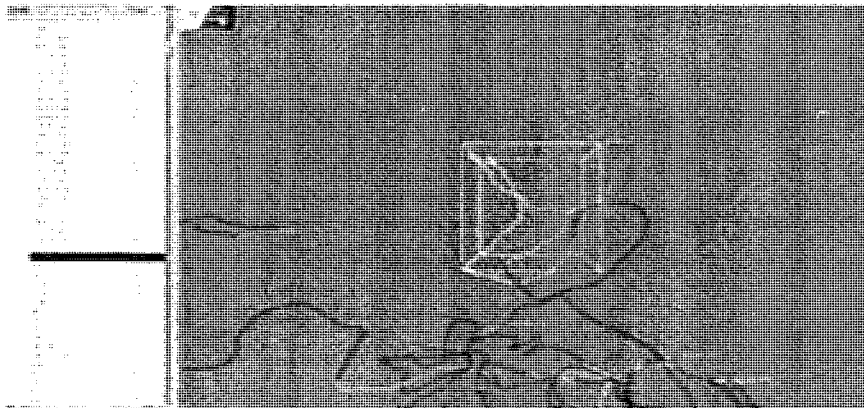
ADN-Viewer (Herisson, 2001) is a bioinformatics visualization tool developed at the LIMSI-CNRS an institute in France, screenshots are shown in Figure 5.1. This is a tool for 3D modeling, stereoscopic visualization focused on virtual exploration and bioinformatics analysis of genomic sequences. The tool provides 3D visualization of the DNA sequence, represented in the form of text as well as a 3D structure model of the naked DNA.



a) 3D visualization of genes (Zoom Off)



b) 3D visualization of yeast (Zoom On)



c) Grayscale screenshot of selecting a particular gene

Figure 5.1: Screenshots from ADN-Viewer

In the following, we describe the more important aspects of this case study in detail along with our analysis of the results for the proposed comprehension criteria.

5.1.2 Evaluating ADN-Viewer

To evaluate ADN-Viewer in terms of its ability to support comprehension, our hypothesis was that users' task performance should depend on the comprehension support provided by ADN-Viewer as assessed by our criteria. To verify this premise, we analyzed the results of a controlled experiment that was carried out with 11 participants from the field of bioinformatics. Based on their different characteristics like – education level, bioinformatics domain knowledge and experience, and skill level with bioinformatics visualizations, we categorized 3 of them as novice, 5 as intermediate and 3 as expert users. Before the actual experimentation, all the participants had a training session, where they were trained to freely explore the tool to make them at ease in using it. A pilot study was conducted with one of the evaluators who explored the ADN-Viewer for the first time.

5.1.3 Experimental Procedure

During the experiment, the participants were asked to perform the following two tasks that were recognized as the most important tasks by the developers of the tool -

1. Group a set of sequences according to similarity in their 3D structures.
2. Find a pair of genes that are spatially close to each other but are far in the textual sequence.

The experimental procedure for each user is outlined in Figure 5.2. Experiments were run one at a time in order to observe the participants using Morae remote viewer tool. In the study, each user experiment lasted between 30-40 minutes.

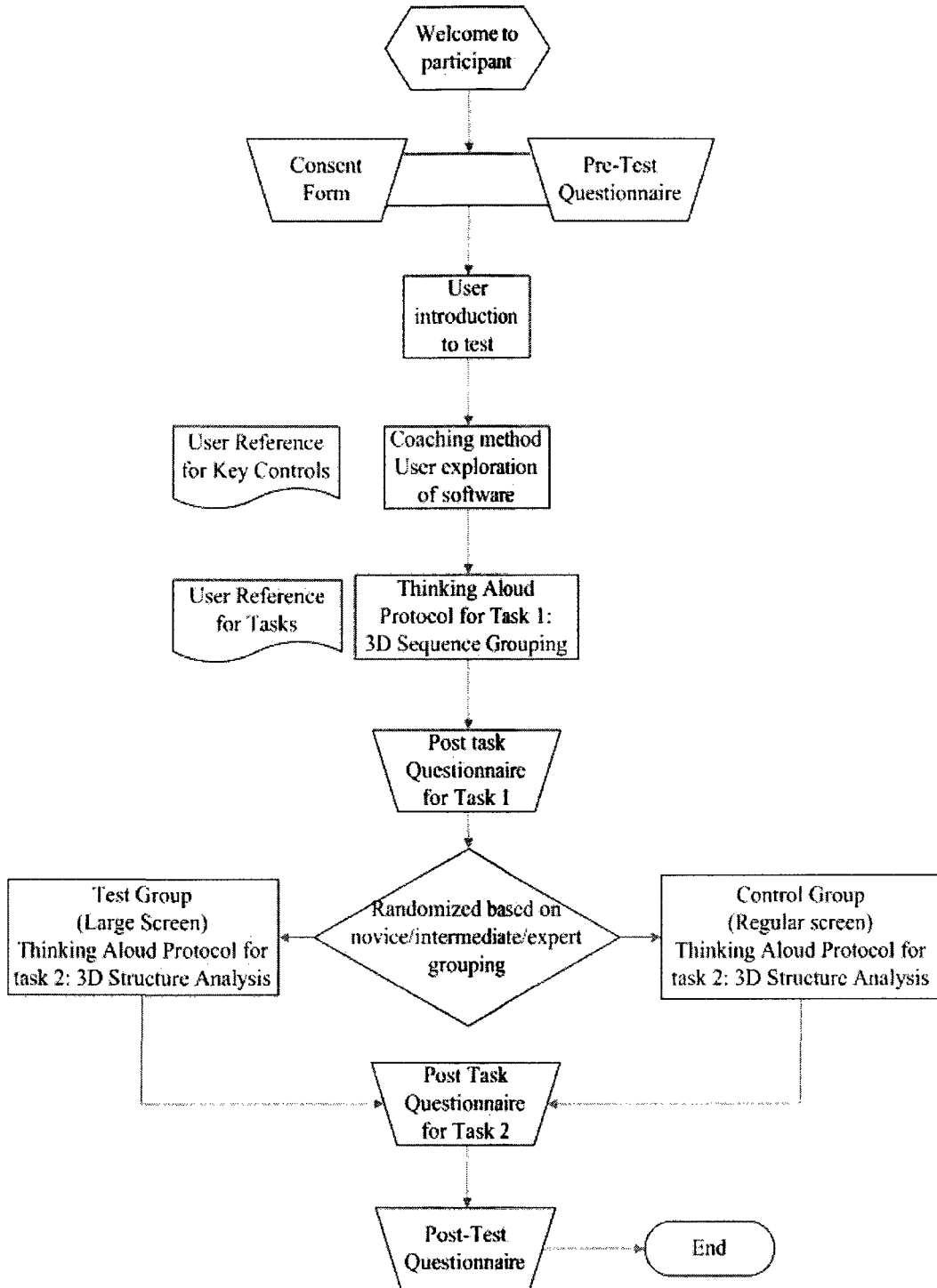


Figure 5.2: Test Protocol for ADN-Viewer

In addition to the pre-test questionnaire, participants were asked to answer from three questionnaires in total – first after task 1 (comprising 4 questions), second after task 2 (a set of 14 questions) and third (18 questions in total) to assess their overall experience with the tool including any comprehension difficulties. The pre-test and post questionnaires were proposed by Joshi (2005) and can be accessed in her thesis. Thinking aloud protocol was used in the study where the participants were asked to speak aloud their thoughts, opinions, emotions and sentiments about their experience of using the system. To get an idea of the comprehension difficulties, the actions of the users/participants during their interaction with the visualization system were also recorded. The strategies to record were the notes of the evaluator, and the voice and video recording of the participants with the Morae tool.

In the following, we describe the results obtained through this case study for our proposed comprehension criteria.

5.1.4 Experimental Results

Based on our analysis of audio and video recordings along with the post-test questionnaires, Table 5.1 shows the combined participants' comments on the comprehension issues in ADN-Viewer. Here, we have categorized the users' verbal findings expressed using a think aloud protocol during the experiment with ADN-Viewer according to the concerned comprehension criterion. Table 5.2 shows the actual results obtained on analyzing the answers to the questions for each criterion. The users' rating depicted for each criterion in this table is the maximum value in percentage (i.e. excellent value on Likert-scale) assigned by all the participants. On averaging the responses for all

criteria (i.e. Sum of all the scores / Number of criteria), we found that comprehension score for ADN-Viewer is 64.18%.

Table 5.1: Applicability of Criteria to ADN-Viewer

Criteria	Applicability to ADN-Viewer
Reachability	ADN-Viewer provides a navigational mechanism to explore a DNA sequence from all directions. However, on finding a particular gene, it is not possible to see its location in overall context.
Simplicity	The view is simple displaying only two colors for genes and inter-gene regions. The icons/labels on the interface are also understandable.
Clarity	The 'start' and 'end' used to depict the start and end of the gene is not clear and causing misinterpretations. It is also not consistent with what is used in bioinformatics domain (where 3'' and 5'' are used rather to depict the start and end of the gene). The participants expect to see the structure of the DNA in the form of A, T, C, G nucleotides rather than the thin line in 3D space displayed using ADN-Viewer.
Distinctiveness	3D DNA structure trajectory is displayed on a black background with yellow color for genes to add distinctiveness and violet for inter-gene regions.
Emphasis	A selected gene is highlighted by a bounding box and it indicates the boundaries of the gene.
Affordance	It is hard to select correctly a particular gene from a cluster of many genes, as the lines used to depict the genes in 3D are very thin. Clicking on appropriate gene is not an easy task in a cluster. The operations for the devices usage are perceived correctly by the participants before actually exploring the system. In addition, participants desire to see perceived affordances to the mouse pointer like showing a hand, grabbing the sequence etc.
Dynamism	Clicking on one gene create the animation effect of zooming onto the selected gene. It is causing confusion as a result of losing context by rotating the entire sequence.
Appearance	The depth factor and other gestalt laws, observed through stereo goggles, are helping in accomplishing task 1 to arrange the DNA's according to matched structure.
Legibility	Displaying all genes names is not legible and view gets cluttered. Comparing all the 3D sequences together is troublesome as number of genes increases.
Perspective-ness	The classical and perspective views provide different viewpoints of looking at one 3D DNA structure and are helping in understanding the structure of genes.
Mapping	Keyboard and mouse actions are natural, except for zooming in and out of 3D DNA structure that can be made easier by just scrolling the wheel without having to press it.

We also observed that first task was accurately completed by 86% of the participants where they were able to group the sequences having similar shapes based on their geometry. In the second task, a pair of spatially close genes was found correctly by 41% of the participants. On averaging these two task performances, we obtained a score of 63.5% which is quite close to our comprehension value and thus the result supports our hypothesis.

Table 5.2: Users' Responses to Criteria

Criteria	Reachability	Simplicity	Clarity	Distinctiveness	Emphasis	Affordance	Dynamism	Appearance	Legibility	Perspective-ness	Mapping
Users' ratings (in %)	44	82	46	82	100	46	50	67	55	60	74

In this case study, we have observed that task performance of the participants (i.e. the percentage of the participants who were able to correctly perform the two assigned tasks) with ADN-Viewer was approximately equivalent to the total comprehension support as assessed by our criteria. Therefore, based on the results of this case study, we concluded that our proposed set of comprehension criteria is correct and is able to appropriately judge the comprehension support provided by a visualization system to its intended users.

5.2 Verifying Correctness: Case Study 2

The second case study was performed with an art visualization environment called OSMOSE (Davies, 1996), and was accomplished in a collaborative work with colleagues in Concordia's human-centered software engineering group. The details of this case study

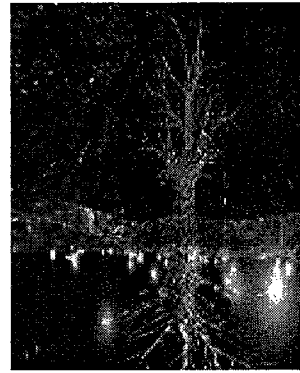
can be accessed in Joshi (2005). A summative analysis of this study is the contribution of the author and is presented below for the purpose of verification of our criteria.

5.2.1 OSMOSE

OSMOSE is a virtual artwork of renowned Canadian new media artist Char Davies. OSMOSE consists of nearly a dozen realms including – tree, forest, pond, subterranean earth, source code and so on, all situated around a central clearing (Davies, 2004). It is an immersive environment where 3D immersion is achieved using a head mounted display (HMD), and interaction is accomplished with the use of body movements and breathing. The artist's conception of OSMOSE is to have an unconscious apprehension of being in a virtual world. Some screenshots of OSMOSE are shown in Figure 5.3.



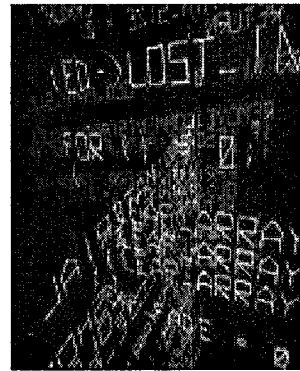
a) Participant using HMD and vest to explore the visualization



b) Tree and pond in the virtual world



c) Forest grid in OSMOSE



d) Code world in OSMOSE

Figure 5.3: Screenshots from OSMOSE (Davies, 2004)

5.2.2 Evaluating OSMOSE

The objective of the study was to understand the applicability of the proposed criteria to the visualization system in order to assess the communicativeness or comprehensibility of the artwork to the participant. In case of OSMOSE, all participants were of varying background and the only task was to explore freely the virtual art environment. 25 participants from the university community participated in the study including administrative staff, professors, undergraduate and graduate students, and some family members.

5.2.3 Experimental Procedure

A different protocol as depicted in Figure 5.4 was used for OSMOSE.

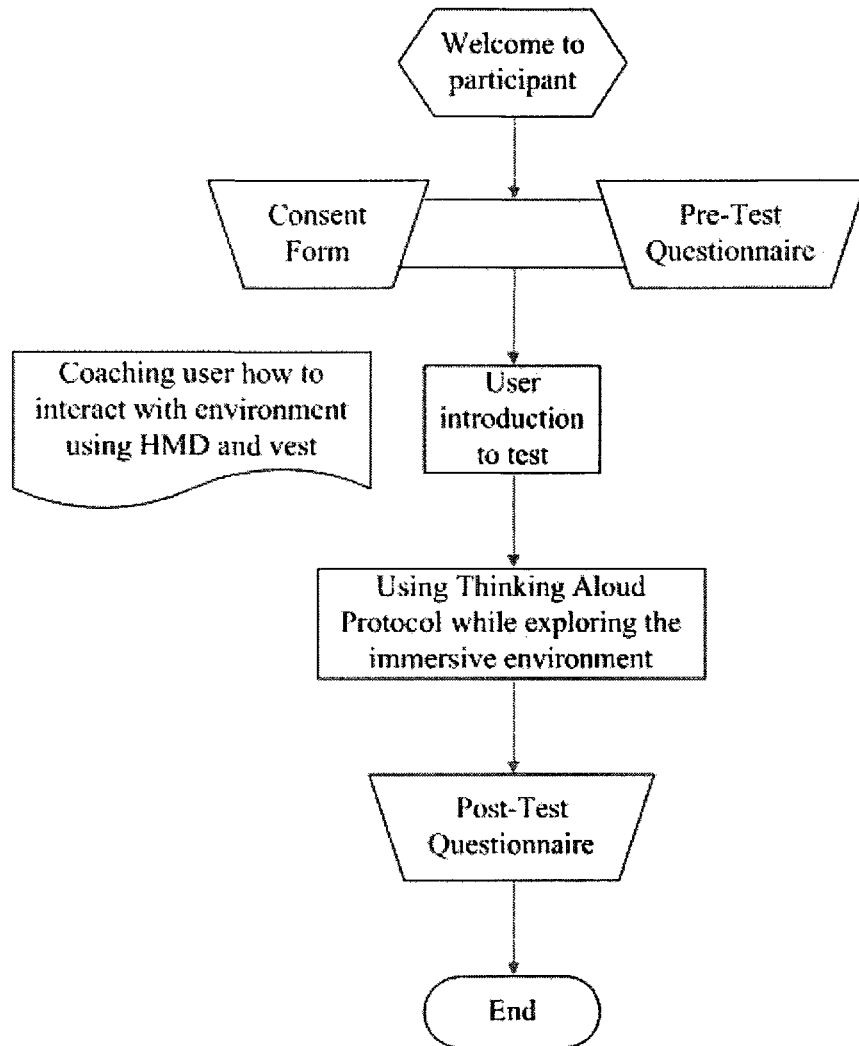


Figure 5.4: Test Protocol for OSMOSE

Here participants from varying backgrounds were initially invited to fill in a user evaluation form, and asked to wear HMD and vest to explore the immersive environment. Then, the participants were taught how to navigate in the immersive environment by moving the head and breathing in or out using the vest. They were asked to explore freely for 15 minutes using breathing actions and body movements. After experiencing

OSMOSE, the participants were asked to fill in a post-test questionnaire to assess different aspects of comprehension. Again, the pre-test and post-test questionnaires were part of Joshi (2005)'s research and can be accessed in her thesis.

5.2.4 Experimental Results

Summarizing the comments of participants in audio and video recordings along with their answers to the post-test questionnaire, we sought the answers to our proposed criteria as shown in Table 5.3. We could not produce a user rating (like in Table 5.2 for ADN-Viewer) for each of the criteria in OSMOSE study, as the post-test questionnaire in Joshi's (2005) work was not described for each of the criteria. However, we were able to obtain a combined score for all the criteria in terms of participants' answer to the following question –

“Were you able to understand what was going in the immersive environment?”

Analyzing the responses from the post-test questionnaire, overall 64% of the users answered yes as a value on the Likert-scale to the answer to this question. Therefore, we may summarize that 64% of the users seemed capable of comprehending the environment using the interaction mechanisms provided in OSMOSE.

In short, through this case study we have found that our proposed set of comprehension criteria was also applicable to assess the communicativeness or comprehensibility of the artwork to the participant based on the assessment of provided features in the visualization system. For all of the proposed criteria, we were able to seek appropriate features in OSMOSE that were signifying some aspect of the user comprehension.

Table 5.3: Applicability of Criteria to OSMOSE

Criteria	Applicability to OSMOSE
Reachability	Navigation depended on user actions- breathing in and out to go up and down, looking around by turning head and leaning forward and backward to move in respective direction. It was not easy to focus on one particular object while viewing other objects in the scene as the environment was moving.
Simplicity	Simplicity was not applied as the designer intention was to give a creative and attractive environment.
Clarity	Many objects were causing different interpretations. Some objects like abyss was not clear to many participants.
Distinctiveness	The objects were made distinctive by using different perceptual attributes.
Emphasis	The main target was the tree being emphasized by its size, position and luminosity.
Affordance	The hardware objects (HMD with headphone afford seeing and hearing, vest afford stretching and loosening). Text afford reading and leaves afford touching. Metaphor of forest, having elements like - tree, leaves, rocks, sounds of birds etc., was used to give the feeling of immersion in the forest. At the same time, there was some inconsistency from real world with no sense of touch or collision detection while passing through objects of the scene.
Dynamism	The environment was moving in the form of moving bugs, moving source code along with the moving world. Most of the time it was difficult to control this movement especially with the breathing metaphor.
Appearance	The Gestalt laws of proximity and continuity were observed by feeling the proximity of leaves and white light sources or bugs emerging "in a line" respectively.
Legibility	The text was legible as the participants were able to speak it loud when they were immersed in source code. The graphics was also interpretable by most of the participants.
Perspective-ness	The OSMOSE environment had multiple views – ground, water, subterranean earth etc. to give a feeling of being immersed in them.
Mapping	Breathing was not natural to feel in immersive environment as the participants were running out of breath. The HMD was heavier for some of the participants.

5.3 Third Iteration - Aftermaths

By applying our criteria to conduct a comprehensive analysis of two different visualization systems, one from scientific visualization category and other from information visualization, we were able to verify the usefulness of our criteria in the practical sense. We describe these criteria basically by observing the user's physical actions to the visualization environment's responses, and by analyzing their assessment of the visualization systems in respective questionnaires. On analyzing the results of two usability studies, we found that each of the proposed criteria contributes partly to signify some aspect of user comprehension. In this analysis, our immediate goal was to seek the features in the visualization systems that enable us to measure the corresponding criteria in some objective manner. This knowledge helped us to derive a general questionnaire to assess each criterion.

The case studies demonstrate the potential benefits of our set of criteria as an important aid to the task of evaluating comprehension of visualization tools/techniques. Our results have shown that our set of criteria could enable evaluators to effectively gauge the comprehension support provided by a visualization system. Therefore, based on our analysis of these two studies, we assume that our refined set of criteria is correct and does not contain any criteria that cannot be assessed using the features in visualization systems. We further verified our criteria for testable property as follows.

5.4 Making the Criteria Testable

To verify if our criteria can be tested, we devised a questionnaire (given in Appendix 'A') comprising of questions for each of the criteria. The questions for each criterion are derived based on the comprehensive literature analysis of their definitions as explained in

previous chapter. These definitions guided us to seek important features in visualization systems. For example, to test the ‘reachability’ in any visualization system, we should look into three main features as follows-

1. the easiness with which users can navigate in the visualization environment,
2. the ability of the visualization system to support undo operations, and
3. the capacity of the visualization system to show current location within an ‘overall context’ of objects displayed in the visualization.

Therefore, we derived three questions to measure ‘reachability’ in any visualization system as under.

1. Are you able to navigate from one location to another in the visualization?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

2. Are you able to undo your manipulation operations (e.g. select, click, move etc.) with the visualization to go back successfully to previously displayed screen?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

3. Are you able to see the location of any information object with respect to an overall context of other information objects in the display?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

We selected a Likert-scale having three values as answers to each of the question; where 'Yes' means 100%, 'Somewhat' is 50% and 'No' is assigned 0% value. Being an ordinal scale, the meaningful statistics that can be applied is the frequency or mode, and the median of the collected responses.

Chapter 6. Software Visualization Systems: A Study on Maintenance Tasks

“Much software is designed and built with little consideration for how it will be used and how it can best support the work its users will be doing.” – Larry L. Constantine & Lucy A.D. Lockwood (1999)

Overview

In this chapter, we study the domain of software visualization (SV) with the intent of subsequently applying our proposed comprehension measurement framework to this domain. A primary application of software visualization is to assist in program comprehension for software maintenance purposes. This chapter presents a thorough empirical investigation of systems in this domain, conducted by us to seek an initial catalogue of software visualization tasks that are required to be fulfilled to accomplish maintenance activities. This task catalogue will be reused in the forthcoming chapters of this thesis.

Specifically, this chapter presents a detailed literature review of software visualization systems, and empirical investigations in related studies by other researchers and practitioners. This chapter also reports on a thorough literature study to prepare a comprehensive catalogue of maintenance tasks that are mentioned in research literature, and are purported to be supported by software visualization tools. We perform an online survey to derive a consistent categorization of software maintenance tasks into traditional maintenance activities. We provide our analysis based on the statistics obtained from the data gathered through this survey, which addresses the categorization and importance of tasks in software maintenance from the viewpoints of experts and intermediates in software maintenance activities. The immediate goal of this study is to be able to identify

the gaps between the needs of software maintainers and the tasks supported by current software visualization tools intended for use in maintenance activities.

6.1 Software Visualizations

Any software system is a pile of code that does something; it is abstract, invisible, and intangible in nature. To seek the meaning of an abstract entity like a software system, we need to represent it in some tangible form. In this respect, software visualization technologies help us by providing graphical representations of various abstractions of the huge source code and ease our perception by giving it altogether a different aspect than that of a source code. Price et al. (1998) defined 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”. Software visualization tries to make the invisible code visible by giving it some form that sheds light on the hidden software structure or meaning of the code. Software structure refers to a collection of artifacts that software engineers use to form mental models while understanding software systems. Artifacts include both software components (e.g. subsystems or packages, classes, interfaces etc.) and also the dependencies among these components (e.g. method calls, data accesses etc.).

Software visualizations are further decomposed into two main categories (Anslow et al., 2004) as under.

1. Static visualizations – these visualizations can be created from either the source or binary files, and provide descriptions of the static elements of source code. They may contain the descriptions of involved packages, classes along with their methods and variables. They also depict the inheritance hierarchies between classes and dependency hierarchies among classes in the source code. The information that is

extracted through these visualizations directly depends on the programming languages or paradigms used for coding. For example, visualizations of object-oriented code give an overview of the classes, methods, attributes, inheritance hierarchies etc. included in the source code; whereas the visualizations of programs written in functional paradigms show the program structure in the form of functions and function calls etc.

2. **Dynamic/run-time visualizations** – these visualizations are created by examining the behaviour of programs during execution to gather the events in a program trace. Various types of information can be displayed with these visualizations like – object creation and deletion, method calls and returns, field accesses and modifications, exceptions, and multi-threading etc.

These two facets of software visualization support many software development activities including analysis, modeling, testing, debugging, and maintenance. Out of these activities, software maintenance is considered rather a heavy and time-consuming activity that involves understanding of complex evolving software systems. IEEE standard defines software maintenance as the process of modifying a software product after delivery to correct faults, improve performance or other attributes, or adapt the product to a modified environment (IEEE, 1998). Software maintenance traditionally involves four basic types or activities i.e. corrective, adaptive, perfective and preventive maintenance (Pressman, 2005). Each activity plays an important role in the evolution and maintenance of any software product and requires a very good understanding of the underlying software system. Although this basic classification scheme of software maintenance has

been enhanced by taking into consideration other objective factors (Chapin et al., 2001), in practice these four conventional activities are the ones most familiar to practitioners. Pfleeger (2001) describes each of the four familiar types of software maintenance activities as follows:

- Corrective maintenance is simply the effort devoted to the removal of defects caused by the routine errors or day-to-day failures. It is needed so that the system complies with the specified performance criteria.
- Adaptive maintenance is required to accommodate the changes occurring in the environment in which the system is used. The most common environmental changes are – changes in the input data, changes in the processing environment (like - installation of a new operating system or an addition of a debugger to enhance a compiler).
- Perfective maintenance activities are carried out to enhance or improve the performance of the software system. It involves making changes to improve some aspect of the system, even when there are no visible errors or failures. There can be major or minor software enhancements depending upon different circumstances. Jones (1998) has described five types of enhancements – block functions, modified blocks, modification and deletion, scatter updates, and hybrid enhancements. A brief explanation for each of these enhancements is as under.

The first type of enhancement is that of adding new features to an application, without the new features causing any extensive internal changes to the original source code. In the second type of enhancement or modified block, it is necessary to make internal changes to the original application in order to attach the additional feature.

These updates are possible only for applications that were originally designed in a very modular, well-structured manner. For the third type, modification and deletion enhancement, the new feature being added to the software replaces a current feature that is actually eliminated. With scatter updates or type 4, several new features are being added at the same time. Type 5 updates or hybrid enhancements come across the classic form of maintenance of poorly structured, aging legacy applications.

- Preventive maintenance is performed for the purpose of preventing the problems before they occur (IEEE, 1998). This is somewhat similar to perfective maintenance in the sense that it involves changing some aspect of the system to prevent failures. However, it usually takes place when one finds an actual or potential fault that has not yet caused the damage.

Visualization tools claim to improve the productivity of software maintainers by providing insights into the invisible code. If this claim is true, then evaluating visualizations should seek to determine how well visualizations aid user' (i.e. software maintainer) comprehension or provide visual insights to invisible code. Therefore, in the domain of software visualization, our research is focused on the evaluation of static software visualizations that assert to provide insights to software maintainers and help them to understand complex software structure.

Towards achieving this goal, we performed literature review on related studies to gather different strategies that can be applied for empirical evaluations/investigations of software visualization systems, and how other researchers have conducted the evaluations of these systems.

6.2 Evaluation of Software Visualizations

In this section, firstly we examine the possible strategies that can be applied for an empirical investigation of software visualization tools.

6.2.1 Approaches for Empirical Investigations in Software Visualizations

Empirical studies result in empirical knowledge or proven concepts that help us to quantify the benefits of software visualization tools. Without measurement in some form, it is very difficult to realize the true value of visualizations to the software community. In a famous book on software metrics, Fenton et al. (1997) suggest that there are mainly three types of strategies that can be used to conduct empirical studies, which are – experiment, case study, and a survey as shown in Table 6.1.

Table 6.1: Types of Strategies (Freimut et al., 2001)

Strategy	Definition
Experiment	A detailed and formal investigation that is performed in controlled conditions.
Case Study	A detailed investigation of a single case or a number of related cases with typical representative projects.
Survey	A broad investigation, where a number of people having experience in a related field participate and present their views on specific issues using standardized forms provided by the surveyors.

According to Fenton et al. (1997) – “a survey is a retrospective study of a situation to try to document relationships and outcomes after an event has occurred... When performing a survey, the evaluator has no control over the situation at hand. The evaluator can record the situation and compare it with similar ones, but he/she cannot

manipulate the variables of the study.” Surveys can be done for descriptive, explanatory and/or exploratory purposes (Wohlin et al., 2000). A survey is appropriate to find out the characteristics, behaviour or opinions of a particular population and to see the differences and commonalities in their responses.

On the other hand, according to Fenton et al. (1997) –“both case studies and formal experiments are usually not retrospective. The evaluator decides in advance what he/she wants to investigate and then plans how to capture data to support his/her investigation... A case study is a research technique where the evaluator identifies key factors that may affect the outcome of an activity and then document the activity in terms of inputs, constraints, resources and outputs. By contrast, a formal experiment is a rigorous, controlled investigation of an activity, where key factors are identified and manipulated to document their effects on the outcome... In an experiment, an evaluator has control and can manipulate relevant variables directly, precisely and systematically.”

Fenton et al., 1997 also suggest various factors that can contribute to a choice of research techniques as shown in Table 6.2.

Table 6.2: Factors Relating to Choice of Research Technique (Fenton et al., 1997, pp: 120)

Factor	Experiments	Case Studies
Level of control	High	Low
Difficulty of control	Low	High
Level of replication	High	Low
Cost of replication	Low	High

As can be seen from the Table 6.2, the key discriminator between experiments and case studies is the degree of control over behavioural events and variables they represent.

A formal experiment is carefully controlled and contrasts different values of the controlled variables; its results are more generalizable. Moreover, controlled experiments are increasingly common in literature as they best enable researchers to rigorously measure and conclusively compare different visualizations (North, 2006). In all, knowledge on the actual effectiveness of the available techniques and tools can be gained only through controlled experimentation (Tonella, 2005).

Due to these advantages of experiments over case studies, we believe that the controlled experiment approach using predefined tasks is the most appropriate to determine the comprehension performance of software visualization tools. Thus, we would prefer to use controlled experiment technique in our research.

We further conducted a thorough literature survey to observe the results of comparable studies as follows.

6.2.2 Relevant Studies

There is still little progress in the evaluation of software visualizations, as most research effort is being spent on the development of novel visualization techniques, ideas and technological innovations rather than judging the current state of SV tools/techniques. Therefore, the field of empirical investigation of software visualization tools/techniques is rather immature and a few researchers have worked informally to characterize and assess the usefulness of these SV tools/techniques. In the following paragraphs, we summarize various related studies conducted by other researchers in the domain of software visualization.

- I. Storey et al. (1996) describe the design and execution of an experiment to assess the usability of three interfaces of a reverse engineering tool. Three game programs of

similar complexity but different sizes were used under study. Twelve users participated in the empirical study, where they were asked to perform eight abstract and concrete tasks with each interface. The users were asked a post-test questionnaire comprising of 20 questions to compare the effectiveness of these three interfaces. The questions were categorized based on five different classes – ‘overall’ to assess user satisfaction, ‘sysuse’ to evaluate interface usefulness, ‘interqual’ to judge interface quality, ‘organization’ to evaluate helpfulness of module organizations in the interface, and ‘confidence’ to determine user confidence in the answers generated by the interface.

2. Storey et al. (1997) performed a user study that compares three tools for browsing source code and exploring software structure. In this study, thirty participants were randomly assigned to *Rigi*, *ShriMP*, *SNIFF* tools and were asked to perform seven high-level program understanding tasks in a controlled experiment. The goals of the experiment were – to observe the strategies used by participants while comprehending program under study, how the tools were supporting this set of preferred strategies, devise a framework to characterize comprehension tools, and provide feedback for tool developers.
3. Bassil and Keller (2001) conducted an online survey of software visualization tools using a questionnaire approach. The online questionnaire was publicized via mailing lists, newsgroups, and emails. The questionnaire was designed using existing taxonomies to extract a list of properties of software visualization tools. The objective of the study was twofold – to assess the functional, practical and cognitive aspects of visualization tools that users’ desire, and to evaluate support of code analysis in the

various existing tools that users' use in their environment. The authors recognized a total of 34 functional aspects along with 13 different practical properties of software visualization tools. They also summarized the cognitive aspects of visualization tools in terms of various usability elements like – 'ease of use', 'effectiveness', and 'degree of satisfaction' etc.

4. Knight and Munro (2001) discuss briefly two main perspectives that should be taken into account when deciding whether or not visualization is effective. These are – the suitability for the tasks that the visualization is intended to support, and the suitability of representation, metaphor and mapping based on the underlying data. They also highlight that domain and data structure has a considerable affect on the effectiveness of any visualization.
5. Pacione et al. (2003) conducted an empirical evaluation of five dynamic visualization tools. The aim of their study was to compare the performance of these tools in general software comprehension and specific reverse engineering tasks. The performance of the tools was judged by conducting a case study with a drawing editor. The evaluation was carried out by a single user who had the knowledge of the drawing editor and dynamic visualization tools. The tools were compared based on four categories – extraction technique, analysis technique, presentation technique, and abstraction level. The questionnaire was divided into two sections – large-scale questions expressing the course of a software comprehension effort, and small-scale questions resembling the course of a specific reverse engineering effort.
6. Storey et al. (2005) proposed a framework for describing, comparing and understanding visualization tools that provide awareness of human activities in

software development. Their framework has five key dimensions: Intent (to capture the general purpose and motivation that led to the design of visualization), Information (data sources that a tool uses to extract relevant information), Presentation (how the tool presents the extracted and derived information to users), Interaction (refers to interactivity of the tools), Effectiveness (determines if the proposed approach is feasible and if tool has been evaluated, deployed). The authors have conducted a survey of twelve software visualization tools and listed the characteristics of these tools with respect to the proposed five dimensions. They have made a number of observations along these dimensions and raised several questions for discussion. They commented that their framework could be used as a discussion tool with software developers, tool designers and researchers.

7. Marcus et al. (2005) conducted a usability study to assess the effectiveness of a software visualization tool named sv3D. The aim of the study was to determine the usefulness and improvement of sv3D as a new technology to support program comprehension. The source program was a documentation software application that was rendered using 3D metaphor of poly cylinders and containers. A total of 35 participants participated in a usability study. The participants were divided into two groups: one group answered the questions using sv3D tool, and other group responded to the questions using tabular data with metrics and source code utilizing the search features in Visual Studio.NET. The answers were analyzed and compared to judge the effectiveness of sv3D tool.

Table 6.3 summarizes the results of these studies on five different dimensions, which are – number of participants, method used for the study, tools used, the program or case study used, and number of tasks involved in the study.

Table 6.3: Summary Chart of Studies on SV Tools

Studies	Participants	Method	Tools	CaseStudy	Tasks
Storey et al. (1996)	12	Controlled experiment	Rigi, Command-Line, Multi-Win, SHriMP	Fish Hangman Monopoly	8 Tasks
Storey et al. (1997)	30	Controlled experiment	Rigi, SHriMP, SNiFF	Monopoly	7 Tasks
Bassil and Keller (2001)	107	Online survey	Over 40 different tools		Four aspects- Functional, Practical, Cognitive, Technical
Knight and Munro (2001)		Expert opinion			Suitability Equation
Pacione et al. (2003)	1	Questionnaire	AVID, Jinsight, jRMTool, Together ControlCenter diagrams, Together ControlCenter debugger	JHotDraw	9 large-scale questions, 6 small-scale questions, and 6 small-scale questions for the case study
Storey et al. (2005)		Experts opinion	SeeSoft, VRCS, Tukan, Advizor, Xia/Creole, Palantir, Jazz, softChange, Evolution Matrix, Augur, Beagle, Spectrograph		Five Dimensions – Intent, Information, Presentation, Interaction, Effectiveness
Marcus et al. (2005)	35	Controlled Experiment	sv3D, VisualStudio.NET	HMS	22 questions

As can be seen from this table, most of the research (Marcus et al., 2005; Storey et al., 1996; Storey et al., 1997) describes the evaluation of the software visualization tools from the perspectives of tool developers only, where they test the tools for a set of tasks that are supported by those tools. That is, these studies attempt to test the effectiveness of software visualizations from their functional viewpoints only but do not attempt to evaluate the comprehensibility of visual information provided by these systems to the actual users. Although, Storey et al. (1996) have used an IBM Post-Study System Usability Questionnaire (PSSUQ) (Lewis, 1995) to evaluate and compare the usability of the interfaces, the questionnaire is not addressing the comprehension features of visualization systems. The questionnaire used by Marcus et al. (2005) to evaluate the usability of their tool is also very specific to their own study only.

Furthermore, most of the independent evaluators like us need to first analyze the research literature on software maintenance and software visualizations to seek the appropriate tasks for further evaluation. Clearly, we need to have a catalogue of software visualization tasks that could be accessed during the evaluation of software visualization systems in general. Towards this endeavour, and to identify the gaps between the needs of software maintainers and tasks supported by current software visualization tools to support software maintenance activities, we conducted a comprehensive literature review as discussed further below.

6.3 Identifying the Needs of Software Maintainers

Many software visualization tools are emerging in research community with the purpose of easing the maintenance of complex software systems. Each tool claims to support a certain set of tasks, related to the information needs of a software maintainer, to

accomplish one or more maintenance activities. However, little is known about what may constitute a comprehensive set of tasks for a maintenance activity. Activities are further broken down into sub-activities (Swanson, 1976; IEEE, 1998) in their definitions. Therefore, it becomes important to have a catalogue of maintenance tasks that are required and should be supported by the SV tools. It is through this catalogue that the tool evaluators could empirically investigate the claims of the visualization tools of being able to provide insights into code for comprehension and support of maintenance activities. This is because if these tools are not designed to be capable of supporting these tasks then their usefulness to the software maintainers is questionable.

Visualization tools for software maintenance are developed in order to fulfill the information needs of the software maintainers and therefore, the user tasks that these tools support are linked to or are derivable from the typical and elementary information needs of software maintainers as shown in Figure 6.1.

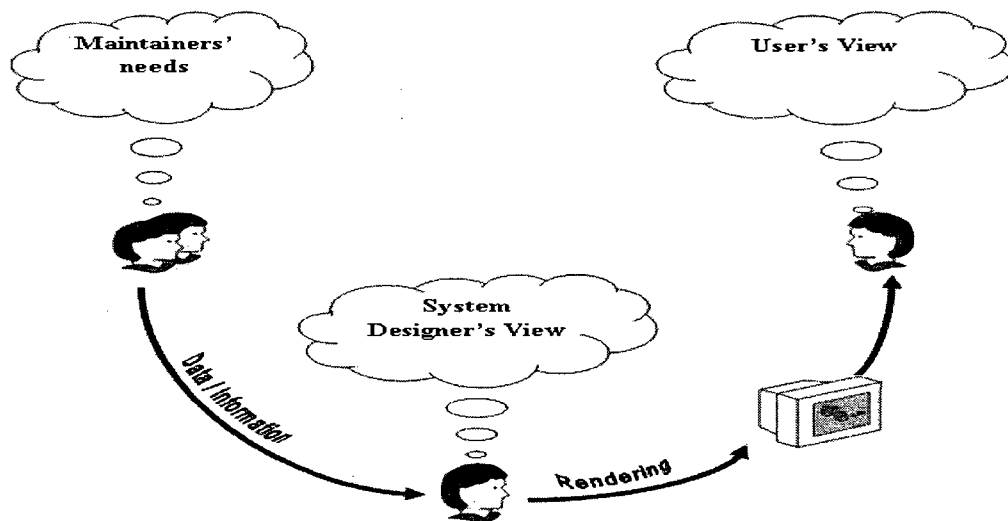


Figure 6.1: Conceptual Views

The system designers, using the available hardware and software resources, transform some of these maintainers' needs into the visualization tool support that is then used by

the intended users. The end users are not necessarily the domain experts who possess background expertise in the maintenance of complex software systems. In Figure 6.1, we can see three different conceptual views of the persons involved in the formation and utilization of software visualizations –

- the first one is of the original software maintainers who have specific maintenance needs to perform maintenance activities and perceive software visualization systems to be capable of fulfilling the desired needs,
- the second view is of the system designers' who are constrained by the existing hardware and software technologies and think of software visualization systems capable of performing a subset of the required functionality, and
- the third view is of the end-users who utilize these visualization systems based on their perceptual and cognitive capabilities and try to infer the design intent along with the interactions to explore the visually represented information.

To be effective, visualizations must match the information they provide with the needs of their users (Cox et al., 2005). Therefore, before conducting the user' evaluation of software visualization tools, the foremost step is to determine the gap between the information needs of software maintainers and the user's tasks supported by the SV tools for software maintenance. We have noted and as also pointed out by Knight and Munro (2001), the software engineering community would benefit from a clear indication of what kinds of tasks are being supported by current software visualization tools. In this regard, we have collected descriptions of a comprehensive set of maintenance tasks that are mentioned in published literature, and are also supported by many currently available software visualization tools.

6.3.1 Current Work

Von Mayrhauser et al. have performed an extensive research (Mayrhauser and Vans, 1997; Mayrhauser et al., 1997; Mayrhauser and Vans, 1998), on understanding the needs of professional software maintainers during the maintenance of large-scale software, using observational field study technique. They have examined the traditional classes of software maintenance quite thoroughly and have identified the information needs of software maintainers to accomplish their tasks. Their categorization of comprehension tasks to individual tool capabilities is described in terms of their three models for code comprehension, is very general and encompasses a variety of tool support including software visualization tools for software maintenance.

Chapin et al. (2001) have proposed a fine-grained classification of the types of software evolution and software maintenance based on the objective evidence of maintainers' activities observed in the software, code, and customer-experienced functionality. They have organized the activities based on four general clusters, which are – support interface, documentation, software properties, and business rules. This classification is again very general, and it includes a variety of factors having impact on the overall maintenance and evolution of the software.

Koskinen et al. (2004) have further studied the 24 most frequent information needs of software maintainers as presented in the empirical studies by Mayrhauser et al., and have discussed the four information sources (i.e. source code, code execution, documentation and other written material, and session history) from where the needed information may be attained. Their classification of the information needs is based on those sources.

6.3.2 Our Perspective

In contrast to these previous studies, our work is more specific; as we are seeking the information needs of software maintainers from the viewpoint of software visualization tools only. The needs are described in terms of the tasks that are required to be supported by the visualization tools for software maintenance. Our list of needs is actually based on a literature review of the work of other researchers in software visualization.

Table 6.4 shows the results of our literature survey conducted to determine the software maintenance tasks required and/or supported by current software visualization tools. We have studied thoroughly the tasks supported by current static software visualization tools (which presently form the scope of our research); at the same time, we also tried to seek other tasks that are mainly supported by dynamic visualization tools.

The tasks listed in Table 6.4 are concrete tasks that are commonly carried out by software maintainers to attain larger maintenance goals of fixing errors or understanding the complete software structure. This list is not exhaustive as the tasks identified above are mainly intended for support by static software visualization tools, which makes up the focus of our present investigation. The table does however provide a comprehensive catalogue of maintenance tasks that may be supported by any static software visualization system. We believe that it could act as a starting point for the development of a comprehensive standardized catalogue of maintenance tasks that can be added to, updated, and maintained for wider use.

Table 6.4: Identified Tasks Along With Their Purpose and Supporting Tools

No.	Tasks	Purpose	Examples of supporting visualization tools / environments
1	Get the execution trace of source code.	The dynamic analysis of the source code gives an insight to determine the source of errors and performance bottlenecks.	TraceVis (Deelen, 2006), Jive (Cattaneo et al., 2004), VET (McGavin et al., 2006), Evospaces (Alam and Dugerdil, 2007)
2	Get the static structure of the software system (Systä et al., 2001; Pacione et al., 2004).	To know class descriptions along with their methods and variables, inheritance hierarchies between classes and dependency hierarchies amongst classes (Anslow et al., 2004).	SA4J (Iskold et al., 2004), Creole (Callendar, 2006), Evospaces (Alam and Dugerdil, 2007), Structure101 (Chedgey, 2007)
3	Find the location of desired code segment (Mayrhauser et al., 1997).	To seek the location of problematic code segment or the segment that needs modification.	TraceGraph (Lukoit et al., 2000)
4	List of all artifacts that call a specific artifact (Mayrhauser et al., 1997).	Call graph display to know what other artifacts are effected by the problematic artifact (Koskinen et al., 2004).	TraceVis (Deelen, 2006), Rigi (Storey et al., 1997)

Table 6.4 (continued)

No.	Tasks	Purpose	Examples of supporting visualization tools / environments
5	Determine the impact of change without having to do it first (Pacione et al., 2004; Iskold et al., 2004) / Ripple effect (Koskinen et al., 2004).	To see what the result of a change made to the software system will have on the rest of the software system. This is required to see the result of removal of the problematic artifact on other good artifacts (Koskinen et al., 2004).	SA4J (Iskold et al., 2004), Structure101 (Chedghey, 2007), Jinsight (Pauw and Vlassides, 1998)
6	Does the run-time behaviour contain regular repeated behavioural patterns (Systä et al., 2001)?	This is required to investigate patterns of repeated behaviour in the system's execution (Pacione et al., 2004). Repeated patterns are the source of common concerns or aspects that can be refactored to improve the software code.	Jinsight (Pauw and Vlassides, 1998), TraceVis (Deelen, 2006)
7	When was an exception thrown or when did an error occur (Systä et al., 2001)?	Information about thrown exceptions is essential for understanding the unexpected behaviour of a target software system (Systä et al., 2001).	Shimba (Systä et al., 2001), Jlint (Artho and Havelund, 2003)
8	Find the location to insert a new artifact.	Location of where to put changes (Koskinen et al., 2004).	SA4J (Iskold et al., 2004), Creole (Callendar, 2006)

Table 6.4 (continued)

No. Tasks	Purpose	Examples of supporting visualization tools / environments
9 Add an artifact and dependencies (if any).	Adding a new artifact along with dependencies is a fundamental task that is required during adaptive and perfective maintenance. The visualization needs to be roundtrip, so that <i>adding or modifying an artifact in the visualization itself</i> should reflect the addition or modifications in the code respectively (Charters et al., 2003).	
10 Find an artifact that is not used (Storey et al., 1996; Systä et al., 2001).	Dangling or orphaned code segments (dead code) that are not used and have no pointers to other code segments need to be removed during software maintenance.	Bauhaus (Raza et al., 2006)
11 Find an artifact that is heavily used in the execution trace or static structure of the software system (Storey et al., 1996).	Based on the number of interactions/ message traffic of other artifacts in the execution trace or on the number of relationships with other artifacts in static structure, locate the heavily used artifact. This is required in order to improve that artifact to achieve greater software system's performance.	TraceVis (Deelen, 2006), SA4J (Iskold et al., 2004)

Table 6.4 (continued)

No.	Tasks	Purpose	Examples of supporting visualization tools / environments
12	Determine which clusters of objects are closely related to one another, based on the amount of message traffic between them (Iskold et al., 2004).	It is needed to improve modularization and refactoring the software systems in appropriate aspects for greater understandability. It is also appropriate in case of knowing the impact of changing any object in the cluster on other objects.	SA4J (Iskold et al., 2004)
13	Find identical coding pattern segments within the source code (Jin, 2001).	Pattern matching to identify the identical coding pattern segments or “aspects” within the source code.	LSEdit (Kapsler and Godfrey, 2006)
14	What is the load on each component of the software system at runtime (Systä et al., 2001; Pacione et al., 2004)?	It is required in order to determine the performance of each object. Runtime load can be measured in a number of ways including memory or CPU usage, object population, or method call frequency (Pacione et al., 2003).	Jinsight (Pauw and Vlissides, 1998)

Table 6.4 (continued)

No.	Tasks	Purpose	Examples of supporting visualization tools / environments
15	History of past modifications (Koskinen et al., 2004).	For evolving software systems, it is important to know how many modifications have been made or how many versions have been released.	CodeCrawler (Lanza, 2004), Structure101 (Chedgey, 2007)
16	Nesting Level of a particular method (Koskinen et al., 2004).	Determine the location of method within the inheritance hierarchy in order to judge the structural complexity.	Creole (Callendar, 2006), Structure101 (Chedgey, 2007)
17	Where in the software system are the hotspots to add additional functionality? (Pacione et al., 2003) (it is included in type 1 enhancement by Jones (1998))	Hotspots are points in a software system where the system designer intends for extensions to be made. These are already defined places in the software, which are left for future enhancements by the system designers.	HotSpotter (Flores et al., 2005)
18	Modify an artifact and dependencies (if any) (type 2 updates from Jones (1998)).	To make internal changes to existing artifacts in order to adapt the system or enhance/improve the performance.	Creole (Callendar, 2006)

Table 6.4 (continued)

No. Tasks	Purpose	Examples of supporting visualization tools / environments
19 Delete an artifact and dependencies (if any) (type 3 enhancements from Jones (1998)).	To delete some artifact in order to improve some aspect of the system, even when there are no visible errors or failures.	
20 Find an exact location to set a breakpoint (Mayrhauser et al., 1997; Koskinen et al., 2004).	In order to reduce amount of run-time information, breakpoints are needed to start and stop recording events during the program execution. This is done to split the event trace into manageable chunks so as to examine only interested parts of the source program.	Shimba (Systä et al., 2001)
21 Find all artifacts that directly or indirectly depend on artifact “A” or Find all artifacts on which artifact “A” directly or indirectly depends (Storey et al., 1996).	A dependency analysis is performed to determine the reliance of system components on other internal or external components (Jin, 2001).	SA4J (Iskold et al., 2004), Creole (Callendar, 2006), Structure101 (Chedghey, 2007)

We followed this literature driven compilation by conducting an online survey of these identified tasks in order to get an independent opinion of practitioners and researchers on various software maintenance tasks and the available visualization support. The details of this survey are as follows.

6.4 A Survey-Based Empirical Investigation on Visualization Support for Software Maintenance Activities

6.4.1 Survey: Rationale

This survey is notably motivated from the work of Mayrhauser et al. to identify the needs of software maintenance professionals during the corrective, adaptive and perfective maintenance of large-scale software. They offer us an interesting basis for further analysis to categorize these maintenance activities into maintenance tasks or needs of software maintainers. The needs are described in terms of the tasks that are required to be supported by the visualization tools for software maintenance. Using this survey technique, we wanted to see the difference and commonalities in terms of tasks among the traditional activities (i.e. Corrective, Adaptive, Perfective, and Preventive) of software maintenance as perceived by the practitioners. The survey also illustrated how the practitioners' perception of these tasks varies with their experience in the domain of software maintenance.

6.4.2 Survey Methodology

This section presents the main phases of our survey.

1. Establish objectives

To conduct a survey, we established two main objectives as follows –

- To categorize the identified tasks into conventional maintenance activities, and
- To rate the tasks in order of their importance in fulfilling the software maintenance goals in general.

Our secondary objectives were –

- To classify the tasks in accordance with support by static and/or dynamic visualization tools, and
- To get the feedback of participants on other visualizations tasks not listed in the survey, but are important from the viewpoint of software maintainers.

2. Questionnaire design

To achieve these objectives, we wanted to have independent opinions of practitioners in the field of software maintenance about software visualization tasks, and this was made possible through the publication of an online questionnaire. We sampled our participants based on one simple profile question i.e. what is their number of years in the field of software maintenance? We prepared a short questionnaire, where the participant was asked to answer three basic questions for each of the 21 identified tasks as follows –

1. Required to accomplish which maintenance activity? (Tick all that apply)

Corrective

Adaptive

Perfective

Preventive

2. Apply to which Software Visualization category? (Tick all that apply)

Static

Dynamic

3. Rate the task in order of importance

Not important

Somewhat important

Extremely important

We applied close-ended multiple choice answers strategy because we wanted to make it easier for the participants to answer the survey, taking at most 10 minutes of their time, as it is not easy to ask remote participants to type answers for subjective questions. Moreover, it was also easy to compare and analyze the results afterwards. We also asked one open-ended question where the participants were asked to comment on additional software visualization tasks not listed in the survey.

3. Online implementation of the questionnaire

The questionnaire was prepared using PHP scripting language and tested with Adobe's Dreamweaver web development application tool before publishing it online to the research community. Responses to the questionnaire were automatically stored in a text file. MS excel software was used for the analysis of the responses.

4. The published questionnaire

Appendix 'B' presents the questionnaire along with the informed consent that was published online to the research community.

5. Data collection

An explicit invitation was circulated to various practitioners and researchers working worldwide in the field of software maintenance and software visualizations. Practitioners were selected based on their acquaintance with us and/or our colleagues. Researchers were identified by searching publications in IEEE and ACM digital libraries, along with

Journals on software maintenance where their work was published in this domain. The invitation message was sent to these practitioners and researchers along with their current students. We also requested the developers of software visualization tools.

In total, 162 people were requested during the March 2007. The response rate¹ was 17.28%, with a total of 28 participants who participated in the study. 2 participants had not answered the questionnaire in total and therefore their answers were not appropriate. Excluding these 2 responses, we were left with a total of 26 responses.

6. Data analysis

The ideal approach to carry out an analysis would have been to base it on the concept of persona where various personal characteristics are assessed to determine representative groups of participants from the surveyed population. However, in order to keep the questionnaire answering efforts to the minimum, we have not included questions enabling persona creation in our survey. Based on the number of years of experience in the field of software maintenance/software visualizations, we did get responses that enabled us to create three categories of participants namely experts (high experience), intermediates (medium experience), and novices (little or no experience).

Out of these 26 participants, 17 were identified as ‘experts’ having an experience of more than 3 years, 7 of them were ‘intermediates’ having experience of 1 to 3 years and 2 ‘novices’ were of less than a year experience in software maintenance/software

¹The low response rate is indicative of the general difficulty involved in carrying out survey-based investigations. However, given that the responses were all voluntary from all parts of the world and from different people with different levels of experience and expertise, we believe that that there is no bias and that it is also representative.

visualization field. The novices were students who had completed a course in software maintenance and comprehension.

The answers to the survey questions were analyzed to determine how each category of participants (i.e. experts, intermediates and novices) classify the given tasks into the four software maintenance activities along with how they assign importance to each task in accomplishing software maintenance in general. The results of this analysis are shown in Figure 6.2 and Figure 6.3 for experts, and Figure 6.4 and Figure 6.5 for intermediates. On analyzing the responses of novices, we saw some drastic variations from both experts and intermediates, and we realized that their answers were not reliable. We believe that it was because of their limited familiarity and their lack of experience with software maintenance and also software visualizations.

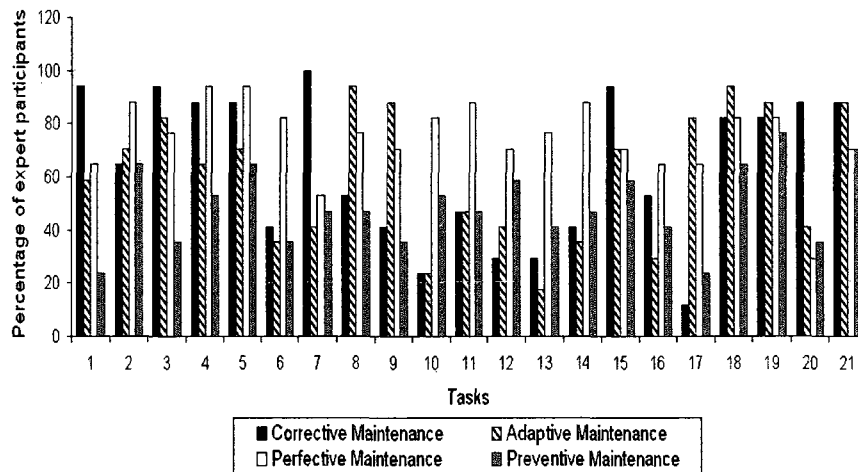


Figure 6.2: Experts' Categorization of Tasks to Maintenance Activities

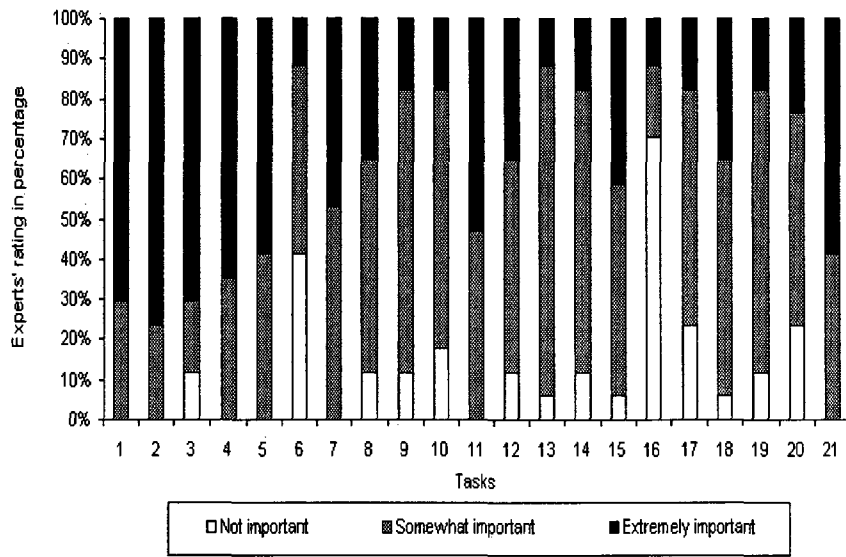


Figure 6.3: Experts' Opinion on Task Importance

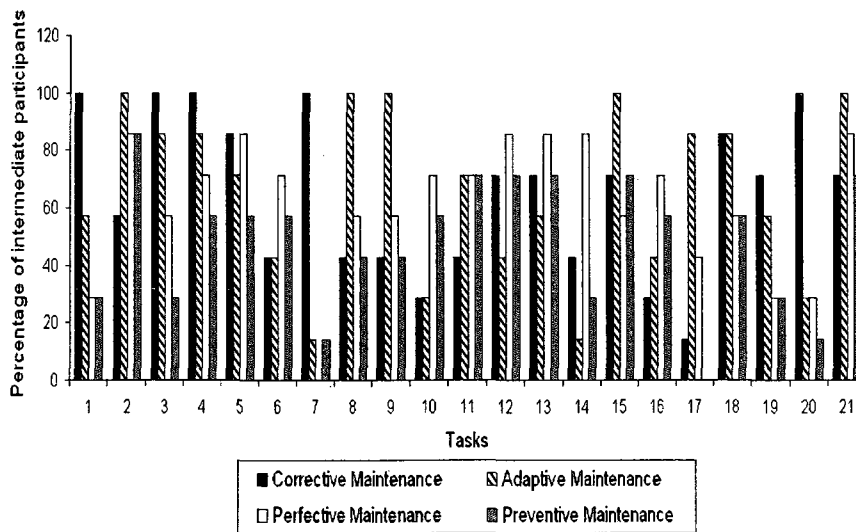


Figure 6.4: Intermediates' Categorization of Tasks to Maintenance Activities

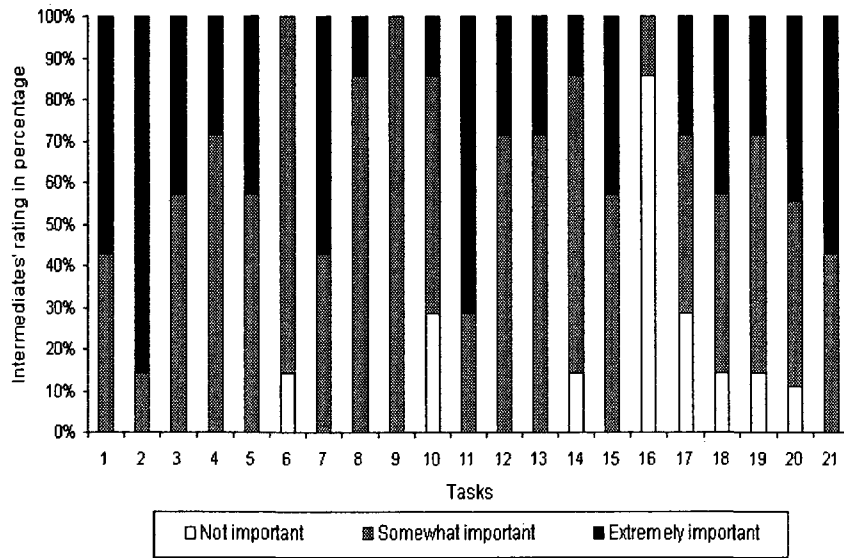


Figure 6.5: Intermediates' Opinion on Task Importance

Looking at Figure 6.2 and Figure 6.4, we can see that task 7 (i.e. when was an exception thrown or when did an error occur?) has been categorized for corrective maintenance by all (100%) the experts and intermediates. Interestingly, we saw in Figure 6.3 and Figure 6.5 that most of the experts (70.59%) and intermediates (85.71%) had assigned 'no importance' to task 16, which was to know the nesting level of a particular method. Task 2 (i.e. get the static structure of the software system) was assigned as extremely important by both experts (76.47%) and intermediates (85.71%) in Figure 6.3 and Figure 6.5 respectively.

The participants were also asked to assign each task to visualization tool category (i.e. static and/or dynamic). Figure 6.6 shows the results of participants' responses. Based on the mean of responses shown by dashed line in Figure 6.6, we can see that tasks 3, 4, 5, 10, 11, 17, and 21 should be supported by both static and dynamic visualization tools, tasks 1, 6, 7, 12, 14, and 20 by dynamic visualizations tools, and tasks 2, 8, 9, 13, 15, 16, 18, and 19 by static software visualizations tools.

On averaging the responses of both experts and intermediates, we derived the categorization of each task in four maintenance activities as shown in Figure 6.7. To classify each of the tasks in terms of maintenance activities, we further divided the participants' responses. Table 6.5 shows the results of this classification where we took those tasks for each activity that are above the 50% range shown through dashed line in Figure 6.7.

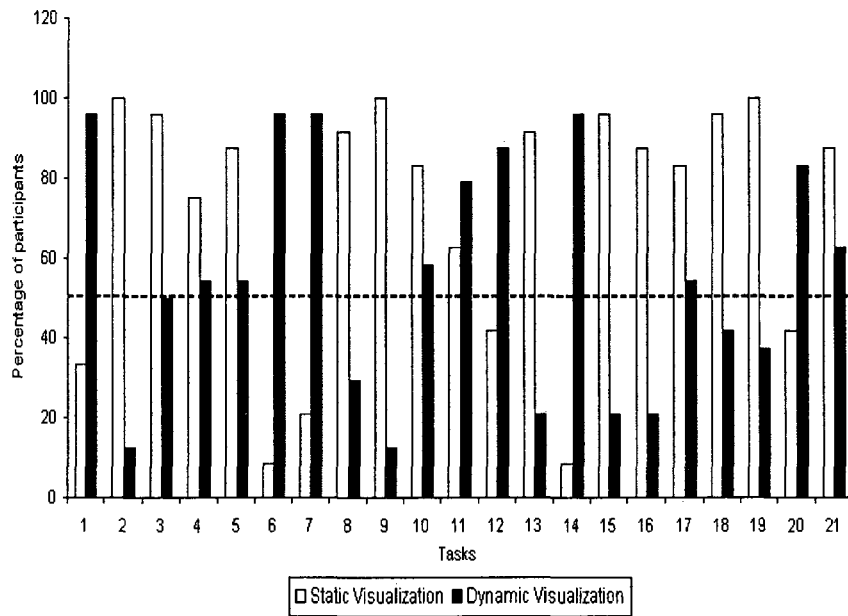


Figure 6.6: Average Opinion on Visualization Category

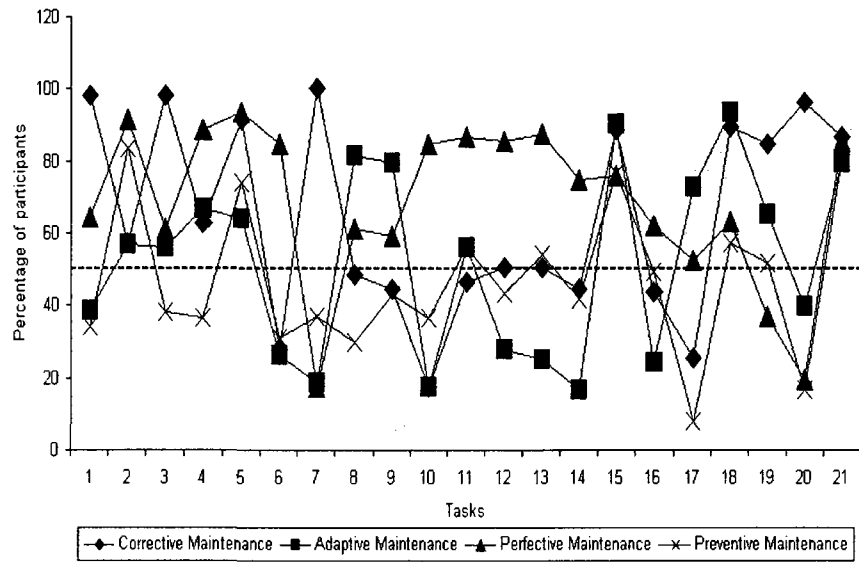


Figure 6.7: Combined Opinion on Tasks' Categorization

Table 6.5: Classification Results

Maintenance activity	Tasks																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Corrective	X	X	X	X	X		X					X	X		X			X	X	X	X
Adaptive		X	X	X	X			X	X		X				X		X	X	X		X
Perfective	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X			X
Preventive		X			X						X		X		X			X	X		X

6.4.3 Discussion

Researchers in the field of software maintenance have given different meanings to these four maintenance activities and there is a lack of agreement on more precise definition of these terms (Chapin et al., 2001). We also agree with this viewpoint, as it is usual to do the same task while performing different maintenance activities as shown in

Table 6.5. Moreover, the definitions given in the literature for some of the tasks are not clear enough as there appears to be incorrect interpretation of these tasks. For example, in our viewpoint many practitioners in our survey seem to have misunderstood task 8 (i.e. find the location to insert a new artifact) and task 17 (i.e. where in the software system are the hotspots to add additional functionality?) and have assumed these to be the same. However, the more correct interpretation is as follows. Task 8 is meant for those additions which are required to fulfill the changing needs not known at system design time. In contrast to this, task 17 points out ‘hotspots’ that were kept by the designers who were already aware of the future needs for system’s expansion. Therefore, logically for the changes that were not previously known, we need to categorize them to adaptive maintenance. In contrast, for the other changes that are kept for expansion or enhancement, we need to classify them into perfective maintenance category.

We also believe that the ‘context of use’ notion is better elucidated through the use of task model. Hence we created a raw task model of the identified tasks using a CTTE (Mori et al., 2002) tool. Figure 6.8 shows the snapshot of the first iteration of this model. This model can be refined further to include other elements of the task model like the interaction, application and user tasks to capture the context of use in which the visualization tool can be used to support the required tasks. Figure 6.8 depicts how the tasks are interrelated, for example – task 9 which is to add a new artifact and dependencies (if any) can be accomplished first by getting the static structure of the software system (task 2) and then we have two choices either to find the location to insert a new artifact (task 8) or to find the hotspots to add additional functionality (task 17) and

on adding an artifact we need to find all those artifacts that depend on the newly added artifact either directly or indirectly (i.e. task 21).

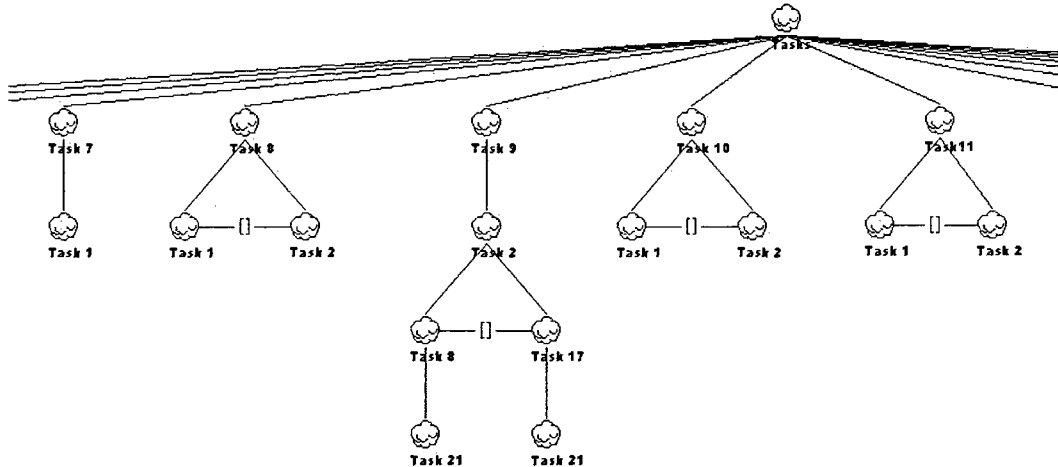


Figure 6.8: Task Model

In response to the survey question to comment on additional tasks not listed in the survey, participants have suggested some literature references (like – Reiss, 2005) to monitoring tasks required to judge the behaviour of the software system like –

- which classes are currently executing,
- which classes are being allocated resources currently, and
- what are the threads in the system and in which state (running, running synchronized, waiting, blocking, sleeping, doing I/O, or dead etc.) each thread is in along with the amount of time spent in each state etc.

In addition, some of the participants have suggested other tasks like –

- finding dependencies between deployed components,
- performance of individual modules in a heavily distributed system, and
- assess impact of mixed technologies in a heavily modular system especially when various modules are managed by different groups.

6.4.4 Conclusion

In this survey, we have highlighted the differences in the definitions of the traditional maintenance activities by classifying them using the maintenance tasks supported by current software visualization tools. This task-based categorization can help to clearly delineate out the ambiguities among the definitions of these activities. Through this survey, we also observed that task descriptions themselves are perceived differently by the software maintainers and need clearer definitions to be interpreted correctly.

We tried to make our questionnaire short by offering a limited number of closed-ended questions. Yet the response rate was low (17.28%). This clearly shows the difficulty in collecting empirical evidences from practitioners in the software engineering community. The list of maintenance tasks is still far from being exhaustive; only some of the currently reported tasks in existing research for static and dynamic visualization tools are listed here for survey purposes.

The proposed classification of visualization tasks in software maintenance activities can guide developers to evaluate their SV tools for respective tasks. We believe that this classification can act as a task model for other independent evaluators like us to empirically investigate the visualization tools for their comprehension and maintenance support. In addition, the identified tasks could guide us to create a task model for software visualization systems that could enable prospective tool developers to build appropriate functionalities in their tools.

Chapter 7. Visualization Patterns: A Context-Sensitive Tool to Evaluate Visualization Techniques

"...encapsulate the concept that varies" – Erich Gamma, Gamma et al Design Patterns

Overview

In this chapter, we discuss the preparatory phase of comprehension assessment by proposing a systematic evaluation mechanism called ‘visualization patterns’ that guides the evaluators and users of visualization systems to compare and understand the functionalities of the underlying visualization techniques. In this chapter, we highlight the need for capturing the problem, context and design solution of any visualization technique in the form of visualization patterns. These patterns are described in two formats – one for evaluators to determine the available task support of visualization techniques applied for the two static software visualization tools under study, and other for the users or participants of the study to understand the capabilities of these techniques in order to explore them during the controlled experiment.

7.1 The Need for Visualization Patterns

Widespread proliferation of visualization tools/techniques has made it difficult for both the users and evaluators to decide on the applicability of a given tool/technique to the visualization problem in hand. The tool users/evaluators have no guidance mechanism that could describe the suitability of visualization tools/techniques to fulfill their objectives i.e. the user/evaluator has no clue if a technique is useful to accomplish his/her required tasks. This is because the context in which a technique can be used dominates the utility of the technique to the user or evaluator. The context describes the main objective of the technique and provides a snapshot of the basic use of the technique.

This context of use is a fundamental and universal characteristic to judge the utility of any software product like - a visualization system. Therefore, we cannot evaluate a visualization technique in isolation without considering the applicable 'context of use'. Sam Uelton (Rushmeier et al., 1995) also says that true quality of visualization can only be measured in the context of a particular purpose, as the same image generated from the same data may be excellent for one purpose and abysmal for another. In the same manner, we cannot say that a visualization technique is universally applicable to all visualization problems. A technique can be good in one context and bad in another. Generalizations made about the observed usefulness of a particular visualization for one task are highly inappropriate as using the same visualization for different tasks often causes the usefulness of the technique to disappear (Casner, 1991). We have to evaluate its' effectiveness in a more abstract manner; by encapsulating the visualization technique in a visualization pattern as mentioned by Wilkins (2003),

“The visualization community has developed a number of techniques that can solve visualization problems that are independent of domain. In effect these techniques are being reused to solve recurring problems. This is the definition of a pattern. Therefore it should be possible to formalize these techniques into patterns”.

Consequently, evaluations should be made in a certain context where the visualization technique can be seen as a solution for a specific problem (c.f. Figure 7.1). The way that we encapsulate a visualization technique with context, solution and problem is apparent to the notion of patterns. We define a visualization pattern as a visualization problem that occurs in a certain context and for which the visualization technique can be a solution. A visualization pattern is different from visualization technique with context, problem, and solution, all made explicit and a rationale provided for the solution.

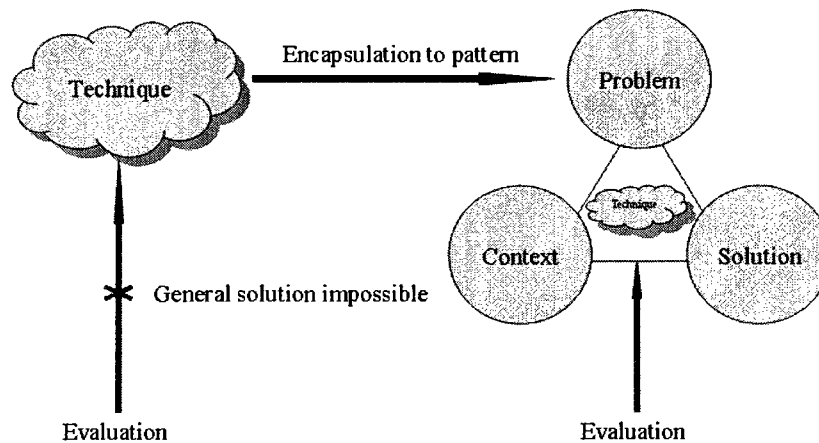


Figure 7.1: Evaluation Strategy for a Technique

A visualization technique could be applied to solve a number of visualization problems (1 to many). For example, TreeMap (Treemap, 2003) is a well known technique that has been applied to solve different problems in a number of distinct domains like – financial (Stock Market TreeMap), bioinformatics (TreeMap Cluster View), information

(NewsMap), business (PeetsCoffee Map), software (Performance Map, DiskUsage Map) and so on. Consequently, a technique has many particular uses, in various domains, with each use being a visualization pattern instance as depicted in Figure 7.2.

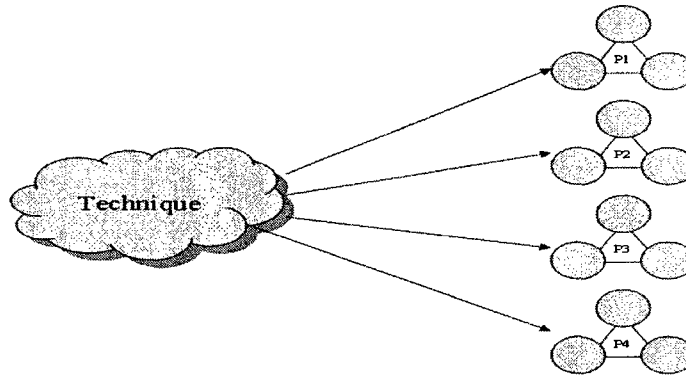


Figure 7.2: Mapping Technique to Pattern Instances

Moreover, we cannot compare any two visualization tools/techniques directly without matching their applicable ‘context of use’. This observation is supported by Tory and Möller (2004), who say “comparison of tools may produce results confounded by the many differences between the tools. Missing or inappropriate features in the test tool or problems in the interface can easily dominate the results...” Each visualization technique has its own limitations. We can only compare patterns when they are similar. Therefore, we can say a technique is better than another only in a certain context.

Patterns also act like a quick reference manual or a short user’s guide that help its’ users to comprehend existing systems. Users may simply refer to patterns in order to understand the capabilities and usefulness of a visualization tool to solve the problem in hand. A preliminary benefit of using patterns is to improve the comprehension of existing visualization systems by getting answers to some probing questions as follows –

- Who can use this technique or in which domain is it relevant?
- What can be done with this technique?

- How can it be used?
- Under what circumstances should it be used?

7.2 Patterns Overview

The concept of design patterns initially proposed by an architect Christopher Alexander for urban planning has been studied and applied in vast research areas. A pattern describes a generic solution to a common problem in context (Alexander et al., 1977). Design patterns have found prominence in many fields including – Software Engineering (Gamma et al., 1995), Graphical User Interfaces (Tidwell, 2005) and Visualizations (Wilkins, 2003). In the visualization domain, Wilkins (2003) has proposed ‘structure’, ‘interaction’ and ‘composition’ patterns for the design of novel visualizations. We differ from earlier work in the sense that former patterns are actually useful for the design of new visualizations, whereas the patterns we are proposing are beneficial for selection/evaluation of current visualization tools/techniques. For the purposes of our study, we encapsulate a visualization technique in a pattern, where we express it in terms of the visualization problem for which it is suitable, the applicable context and the proposed design solution. We have chosen the pattern format as described by Borchers (2000) and have adapted it to our needs for visualization techniques. The general format of a pattern for any visualization technique is shown in Table 7.1.

Table 7.1: General Format of a Pattern

Name or Title	Helps to refer to the pattern's central idea quickly and builds a vocabulary for communication.
Context	The visualization situations in which the pattern can be used.
Problem	A statement of the visualization problem that needs to be addressed.
Forces	The factors which must be considered when applying the pattern under current context of use.
Solution	The proposed visualization design solution to the problem.
Examples	Show existing situations or cases in which the problem at hand can be (or has been) encountered, and how it has been solved in those situations.
Related pattern(s)	Reference to some patterns that solve similar or related problems and to patterns that refine the pattern under describing.

To further test the usefulness of our pattern-oriented evaluation approach, we conducted a case study with the two popular static software visualization tools, IBM's Structural Analysis for Java (SA4J) and Creole as follows.

7.3 Case Study: A Pattern-Oriented Evaluation of Software Visualization Tools

Patterns in general have been used by many researchers (like - Zhu et al., 2004; Berg and Ahlstrom, 2005; Georgiakakis et al., 2006) in different domains as an important aid for an evaluator. In this case study, we are exploring how visualization patterns can guide an evaluator to perform a comparative analysis of static software visualization tools/techniques. The detailed explanation of the case study is as follows.

7.3.1 Objective

The objective of this case study is to evaluate the suitability of given static visualization tools in visualizing the structural dependencies in a software system. The

techniques used in these tools are encapsulated in a pattern format, where the constraints or forces that limit the use of these techniques are described in terms of the tasks that are supported for software maintenance/comprehension and interactions with a software system. The three primary elements of our case study, i.e. – tools/techniques, system under study, and tasks required for evaluation, are described as below.

7.3.2 Tools/Techniques

The tools evaluated in this case study are used for the structural analysis of a software program. Being independent evaluators, we were looking for tools developed by other researchers/practitioners. Unfortunately, most of the tools on Internet (SmallWiki, 2007) are commercial and are not accessible for usability evaluation. Others are research prototypes and are not fully functional. However, we do believe that the tools we have chosen are representative of the kinds of tools that are developed in industry and academia to support software comprehension during software maintenance.

For our study purposes, we have chosen two tools – Structural Analysis for Java (SA4J) (Iskold et al., 2004) and Creole (Callendar, 2006). Each tool uses more than one visualization technique to visually represent the software structure. A detailed explanation of the capabilities and functionalities of each tool is given below:

- **Structural Analysis for Java (SA4J) (Iskold et al., 2004)**

SA4J is a tool introduced by IBM for structural analysis of Java applications. SA4J analyzes the class files in order to show the static structures of Java applications. SA4J measures the stability of an application structure by evaluating the web of dependencies among different objects like – packages, classes, and interfaces of a Java application. This analysis provides quantitative and deterministic evaluation of the application

structure. SA4J provides browsing for detailed exploration of anti-patterns (bad design elements) in the dependency web, and enables ‘what if’ analysis in order to assess the impact of change on the functionality of the application. It also provides a spreadsheet view of various items along with a dependency pyramid view of an application. Here, the basic idea is to represent the software as a pyramid of dependencies; where the objects that do not depend on anything are at the bottom, objects that depend on them are on the second level and so on. SA4J is a standalone system, pre-packaged with Java Run-time Environment (JRE) 1.4.1_01, and can be installed on many platforms like – Windows 2000/XP/NT, Linux, and Sun Solaris 8/9. This tool is freely available and can be downloaded from the host IBM or sourceforge.net site.

- **Creole** (Callendar, 2006)

Creole is an integration of Simple Hierarchical Multi-Perspective (SHriMP) with Java Development Tools included in the Eclipse platform (Lintern et al., 2003). SHriMP is both an application and a technique, designed for visualizing and exploring software architecture, developed by the University of Victoria's Department of Computer Science. This visualization technique is incorporated into the Rigi reverse engineering system which is developed to extract, navigate, analyze, and document the structure of evolving software systems (Storey et al., 1995). Creole adds its own perspective to the Eclipse platform and explores the Java source code visually by displaying its structure in the form of different software objects (packages, classes, interfaces etc.) and the relationships (calls, accesses, extended-by and so on) between these objects. Creole uses five different layouts to provide multiple perspectives of the software structure. In Creole views, the source code is an integral part of the structural documentation, as opposed to opening a

file containing the artifact's corresponding source code in a separate text editor like in many other tools. The relevant source code for software artifacts represented by leaf nodes is displayed directly inside the nodes in these views. This allows the user to browse source code while simultaneously visualizing the location of the code in a software hierarchy. The most recent version, Creole 1.6.1, works as a plug-in for the Eclipse platform, and needs Java to be installed.

7.3.3 Software Program for Analysis

To select an appropriate software system to analyze using the visualization tools under study, a number of factors have been considered. These include – programming domain, program size, complexity, quality, and availability. The detailed explanations for each of these program characteristics are as follows.

- Programming domain

The software visualization tools that are available currently are developed by different developers and have different hardware/software requirements. So, we were looking for a platform independent source language that most of these tools could support. We opted to use Java as a source language for the software program to be analyzed, as both our study tools (i.e. SA4J and Creole) visualize the Java applications. Our experience and comfort level with Java language was another reason for our experimentation with a Java source program.

- Program size

The size of the program is being constrained by the scalability issues of the software visualization tools under study. Scalability concerns the space and time complexity of

visualization techniques, for instance – automatic layouts for large graphs, as well as the need to avoid information overload for the viewer (Koschke, 2003).

- Complexity

In general, simple programs may not have the same need for visual analysis, as they could be comprehended more easily. However, as a software program becomes cumbersome and complex, it necessitates the use of software visualization tools. Broadly speaking, there is not a standard threshold limit for the complexity factor of a software program, i.e. how complex a software program should be so that visualization becomes helpful. For our study purposes, we decided to use a medium size software program, which we consider as being neither too simple nor too complex.

- Quality

Quality of the code is also taken under consideration while selecting appropriate system for study. The source project should be free of bugs or other exceptional errors. This is required in order to compile the code completely and use its byte code or source code to make visualizations with the software visualization tools in hand. Some tools require the class files and not the source files to produce visualizations, and if the source code is not free of errors or is not of good quality then the tool may not create the required visualizations.

- Availability

Availability of the source program is another contributing factor for its selection. We decided to take the source project from the list of open source projects that are freely available.

Based on above considerations, we decided to use for our case study an open source application called BORG (Berger-Organizer), which is a calendar and task tracking system written in Java. The calendar's functionality is similar to that of other personal information managers such as – Microsoft Outlook, Mozilla Calendar, Palm Desktop, Yahoo Calendar and so on (Berger, 2007). The system, with its latest version 1.6, has evolved to a stable system. On the source forge site, this project is ranked at position 3,348, downloaded more than 75,000 times and is described as highly active project having activity rating of 98.33% (BORG ranking, 2007). SA4J gives a summary of BORG as – 99% stable system comprising of 239 objects (i.e. 54 packages, 172 classes, and 13 interfaces) and 351 relationships among these objects.

7.3.4 Tasks

Tasks are a key component to conduct any form of empirical evaluation. Maletic et al. (2002) rightly state that the tasks are the driving force behind a classification of software visualization systems. Having an appropriate task list is a prerequisite before conducting any evaluation. Toward this endeavour, we conducted a thorough literature survey (as explained in chapter 6 of this thesis) to seek the software comprehension and maintenance tasks that are mentioned by other researchers, and are required to be fulfilled by software visualization tools. Our task list that comprises the tasks supported by current static software visualization tools is described below.

- **Maintenance tasks**

To understand the static structure of a software system and perform maintenance during maintenance activities by any static software visualization, typically, following are the tasks that are required to be accomplished.

- M1** Get the static structure of the software system (Systä et al., 2001; Pacione et al. 2004).
- M2** Find the location of desired artifact (Mayrhauser et al., 1997).
- M3** Find an artifact that is not used in the static structure of a software system (Storey et al., 1996; Systä et al., 2001).
- M4** Find an artifact that is heavily used in the static structure of a software system (Storey et al., 1996).
- M5** Find all artifacts that directly or indirectly depend on artifact “A” (Storey et al., 1996; Mayrhauser et al., 1997) or Find all artifacts on which artifact “A” directly or indirectly depends (Storey et al., 1996).
- M6** Determine the impact of change without having to do it first (Iskold et al., 2004; Pacione et al., 2004) or Ripple effect (Koskinen et al., 2004).
- M7** Add an artifact and dependencies (if any).
- M8** What is the history of past modifications (Koskinen et al., 2004)?
- M9** What is the nesting level of a particular method (Koskinen et al., 2004)?
- M10** Where in the software system are hotspots to add additional functionality? (Pacione et al., 2003) (it is included in type 1 enhancement by Jones (1998))
- M11** Modify an artifact and dependencies (if any) (type 2 updates from Jones (1998)).
- M12** Delete an artifact and dependencies (if any) (type 3 enhancements from Jones (1998)).

In addition to the maintenance tasks, we believe that appropriate interaction mechanisms are also required to be provided by any visualization tool. This is because interaction is the basic requirement to help in achieving the maintenance tasks required of

visualization tools. The interaction mechanisms providing for the navigational needs of the users are explained below.

- **Interaction mechanisms**

With current technologies, visualization tools provide a variety of interaction mechanisms to users. Interaction mechanisms allow the users to directly interact with the visualizations and dynamically change the visualizations according to their exploration objectives. According to Knight and Munro (2001), interactions allow users to investigate, browse, and interrogate various aspects of information without relying on predefined fixed views. Visual Information Seeking Mantra by Shneiderman (1996) defines seven basic information seeking mechanisms or interaction techniques that all visualizations should support, and are explained as follows –

I1 Overview: Get an overview of the entire collection of data that is represented through visuals. With large systems, this often results in incomprehensible visualizations. This task can be accomplished by using overview strategies like – ‘overview plus detail’ views i.e. zoomed out views of each data type to see the entire collection in addition to an adjoining detailed view, and ‘fisheye approach’ where the fisheye-lens metaphor is applied by magnifying the objects in the center of the view while reducing the size of objects away from the center. Fisheye views provide context and detail in one view.

I2 Zoom: In general, users are interested in only some parts of the visualization where they want to focus while retaining the global context of the overall visualization. The visualization tools should provide the functionalities to control the zoom-focus and zoom-factor in visualizations. Zooming can be done in one dimension at a time by

moving the zoom bar controls or can be accomplished in 2D by adjusting the size of the field-of-view-box.

- I3** *Filter*: In order to tackle the clutter in visualizations, it is also necessary to filter out uninteresting or unwanted items. Maletic et al. (2002) point out that in order not to disturb the global context by filtering there should also be some kind of abstraction of removed parts.
- I4** *Details-on-demand*: To facilitate understanding of each artifact in the visual clearly, it is also important to get its details on demand. The common approach used to fulfill this task is to simply click on an item to get a pop-up window which shows the values of each of its attributes.
- I5** *Relate*: For a hierarchical data structure, users need to view relationships among items. Users can select an item and then highlight items having similar attributes.
- I6** *History*: A history of the actions performed with visualization should be recorded to support various operations like – undo, replay and do progressive refinement. It helps to tackle the ‘where was I syndrome’ in visualizations, allowing users to go back to their previous state in exploring the visualizations.
- I7** *Extract*: A visualization tool should allow extractions of sub-collections and of the query parameters. This task concerns saving the current state of visualization for future explorations (Maletic et al., 2002).

7.3.5 Case Study Results

Table 7.2 summarizes the results of our pattern-oriented evaluation of these tools, where we have encapsulated the underlying visualization techniques used for each tool in visualization patterns that are named according to the layout styles used in respective

techniques. Here, all the visualization techniques solve a common visualization problem of displaying a hierarchical structure showing dependencies among software objects in a software system. The context or situations in which the pattern can be used is also similar in these patterns. However, the forces and solutions vary, and these significantly differentiate each technique from the other. The forces factor of each pattern is described in terms of supported maintenance tasks and interaction tasks.

Table 7.2: A Pattern-Oriented Analysis of Tools

	Structural Analysis for Java (SA4J)		Creole	
Name or Title	Radial tree	Pyramid	Nested view	Tree
Context	The display consist of number of software objects (packages, classes, and interfaces) and their inter-relationships or structural dependencies in the source code			
Problem	How to display large hierarchical tree structures showing dependencies among software objects?			
Forces	{M1, M2, M3, M4, M5, M6} {I1, I2, I3, I4, I5, I6, I7}	{M1, M2, M5, M6} {I1, I2, I4, I5, I6, I7}	{M1, M2, M3, M4, M5, M9, M11, M12} {I1, I2, I3, I4, I5, I7}	{M1, M2, M3, M4, M5, M9, M11, M12} {I1, I2, I3, I4, I5, I7}
Solution	Use a radial tree representation	Use a skeleton view	Use nested rectangles	Use a standard tree view
Examples	Sunburst, Radviz	Icicle plot	PhotoMesa Image Browser, SmartMoney, NewsMap	Visualize it!
Related pattern	Pyramid, Treemap, Tree, Cone Tree, Explorer	Radial tree, Treemap, Tree, Cone Tree, Explorer	Radial tree, Pyramid, Tree, Cone Tree, Explorer	Pyramid, Radial tree, Treemap, Cone Tree, Explorer

In the following, we describe summative results of our pattern-oriented evaluation of visualization techniques employed in SA4J and Creole by explaining in detail the forces and solution elements of corresponding patterns.

- **Structural Analysis for Java (SA4J)**

SA4J uses two different visualization techniques called ‘radial tree’ and ‘pyramid or skeleton view’ to show the static structure of a software system. The patterns corresponding to these techniques are as under.

- a) **Radial tree pattern**

Radial tree technique displays different software objects like – packages, classes, and interfaces of an application along with their relationships in a radial fashion as shown in Figure 7.3. The idea is that object nodes are placed around the circle and their relationships are shown with directed lines emanating from the source to destination node. With this technique, complete static structure of the software system can be shown very efficiently.

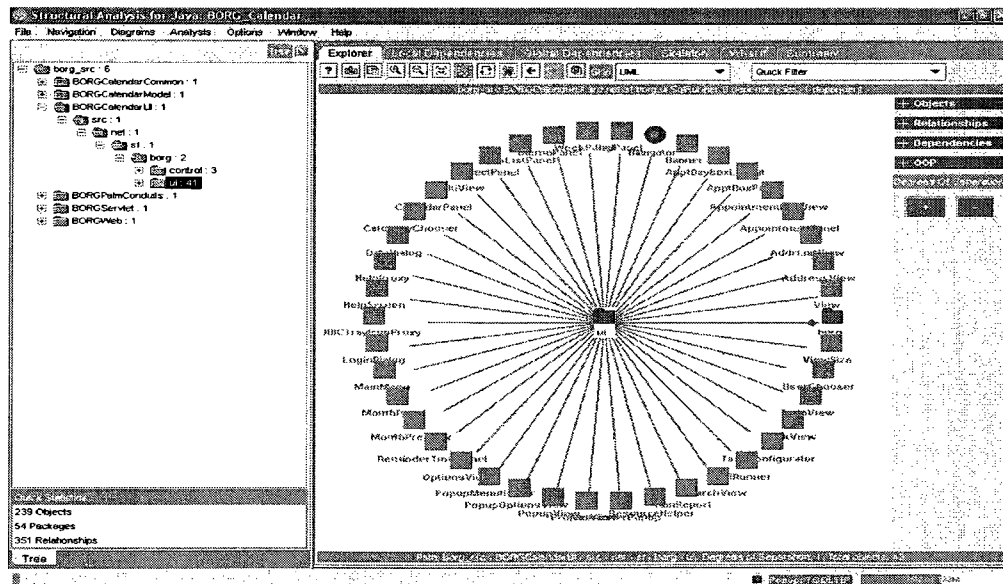


Figure 7.3: Radial Tree Visualization

This technique fulfills all the maintenance tasks from M1 to M6. However, the tasks of adding/modifying/deleting an artifact or dependency in the visual(s) are not possible with this technique. It is because the classes/byte code cannot be altered once visualized

in this tool. Other tasks like finding hotspots or nesting level of a particular method are also not supported by this technique.

This technique is excellent in terms of its interactivity. It supports all the seven interactions tasks to navigate in large static structures effectively. It uses focus plus context viewing, allowing enormous structures to fit within fixed space of computer screen. It provides a fine zooming capability to zoom on a particular node while keeping the neighbouring context intact. A data-tip is tuned with every node in the structure to display details on demand. Filtering, relate operations are also fulfilled with this technique. A navigation history of a total of 30 actions can be accessed to support undo and other operations. This technique also permits one to save specific shots of substructures in jpg or DIR file exchange formats.

b) Pyramid pattern

This technique shows a dependency pyramid view of an application (c.f. Figure 7.4). The basic idea here is to represent the software as a pyramid of dependencies – the entities with only outgoing dependencies on the bottom, those with only incoming dependencies on the top. Each square corresponds to either one object (class/interface/package) or one tangle (set of objects that change together). In this view, a stable system should have a normal pyramid shape. An unstable system may look like an upside down pyramid shape.

This technique did not accomplish tasks M3 and M4 as was done by Radial tree. There is no special visual attribute that could tell the analyst which artifact is heavily used or not used in the static structure. Again, like Radial tree it does not support tasks from M7 to M12.

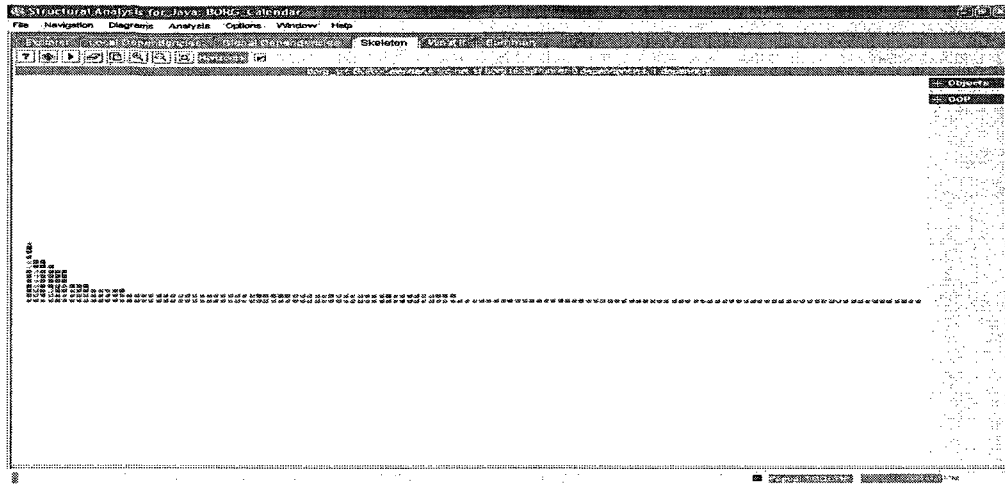


Figure 7.4: Pyramid Visualization

This technique did not support the task of filtering unrelated items, as we were expecting the tool to show the filtered items in the form of pyramid of related items. However, it was illustrating the related items in a radial fashion using the radial technique as described previously. This technique coped quite well with rest of the interaction tasks.

- **Creole**

Creole uses five different layouts to provide multiple perspectives of the software structure. These are – ‘Nested view’, ‘Spring’, ‘Tree’, ‘Radial’, and ‘TreeMap’. “The ‘Nested layout or Grid’ arranges all the children (or sub-nodes) of a specific node to fit into the inner bounds of that node in a rectangular format. It does not take arcs into consideration when laying out the nodes. The ‘Spring’ layout simulates a mechanical system where highly connected nodes tend to be pulled together and more isolated ones tend to be pushed away from each other. The ‘Tree’ layout extracts an acyclic graph from a set of nodes by tracing their relationships. The ‘Radial’ layout positions the nodes and arcs in a radial pattern or format. ‘TreeMap’ layout is a space-filling method of visualizing large hierarchical data sets. It visualizes the hierarchical structure by

representing the nodes with nested rectangles” (Creole User Manual, 2006). For our further usability study purposes, we explored two out of these five visualization techniques as follows.

a) Nested view pattern

This is the default layout in Creole and a screenshot is shown in Figure 7.5. It is based on the standard “contains” relationship, which means that a node contained within another node indicates a parent-child relationship between them. It is a space-filling approach of visualizing large hierarchical data sets (packages, classes and interfaces) and their inter-relationships or structural dependencies in the source code.

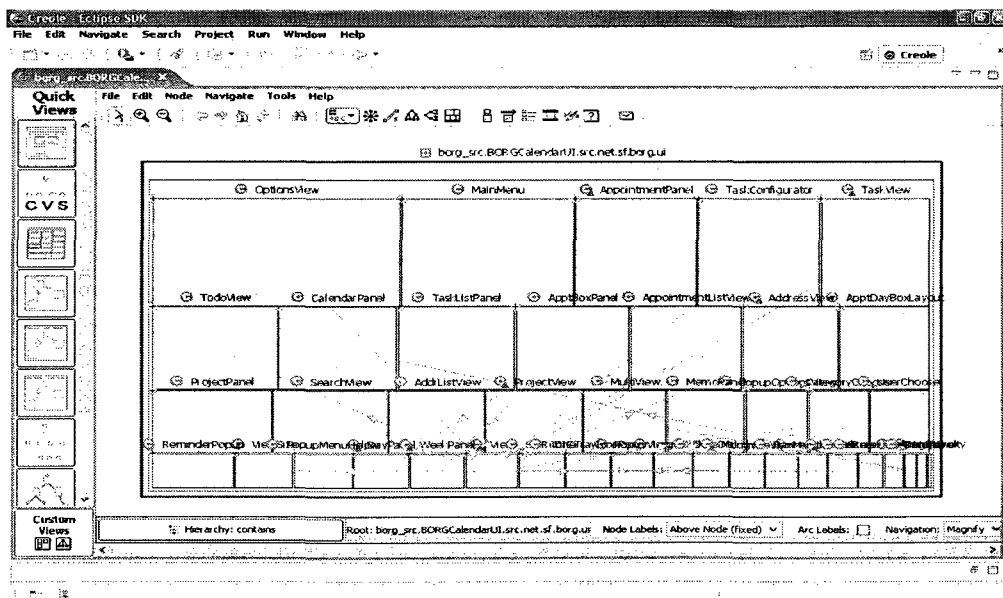


Figure 7.5: NestedView Visualization

This technique performs well with most of the tasks from the studied set. However, it does not tackle the “impact analysis”, “adding an artifact” directly on the visual etc. as shown in Table 7.2. The most interesting feature of Creole views is that source code for software artifacts is displayed directly inside the nodes. This allows the user to browse source code and make changes simultaneously along with visualizing a software

hierarchy. This technique presents software structures using fisheye views of the nested graphs. The fisheye-lens metaphor magnifies the nodes of interest in the graph while concurrently shrinking the remainder of the graph. This technique also provides a mechanism for presenting details of a large information space while also displaying contextual cues at the same time. The history mechanism is not supported properly by this technique. Although, there are forward and backward buttons on the interface itself, they are of no use once you alter the location of nodes within the visual. The image can be exported in formats jpg or png. The snapshots can also be saved on a filmstrip.

b) Tree pattern

This is another layout that is provided by Creole and is shown in Figure 7.6. Nodes may represent software artifacts, and edges may represent semantic relationships among those artifacts.

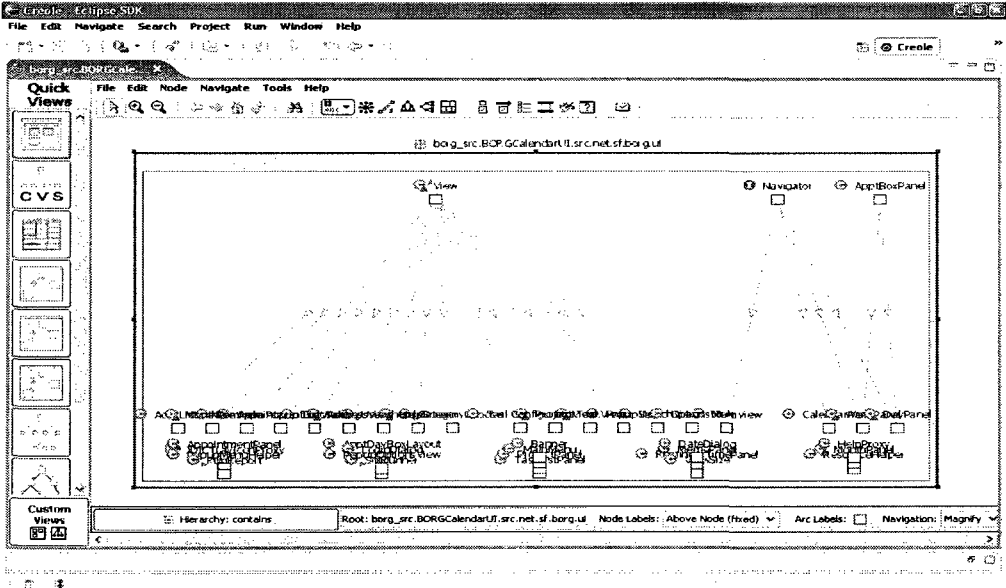


Figure 7.6: Tree Visualization

It is based on the general metaphor of a tree where branches from the root node(s) emanate to child or leaf node(s) and so on until the complete hierarchy is formed. This is

a very basic technique of depicting the hierarchical structure; however, it can easily become clumsy with large structures. As shown in Table 7.2, this technique supports the same set of tasks as are carried by nested view technique.

7.3.6 Discussion

A big gap between desired tasks and the tasks supported by SA4J and Creole is observed. As can be see from Table 7.3, no tool is able to address all the listed maintenance tasks. Both the tools provide very good interaction mechanisms, with radial technique superseding all other techniques. We must remark here that the tasks' support does not imply that Creole is more effective than SA4J in fulfilling the common visualization problem of displaying the static structure. This classification through patterns is a first step to empirically assess the value of these visualization techniques to the ultimate users. The actual effectiveness of these tools/techniques can only be judged through usability evaluations with the real users and real experiments. During our analysis, we have seen that SA4J is more efficient than Creole based on the response time while exploring the visual(s). Proper usability assessments of these tools can further tell us the effectiveness of these tools.

Table 7.3: A Comparative Summary of Tasks

Tools	SA4J		Creole	
	Radial	Pyramid	NestedView	Tree
Maintenance tasks (12)	6	4	8	8
Interaction tasks (7)	7	6	6	6
Total task support (19)	68.4%	52.6%	73.7%	73.7%

7.3.7 Conclusion

This case study, a comparative analysis of SA4J and Creole, demonstrates the benefits of using visualization patterns as an important aid for an evaluator in the task of appropriate selection and evaluation of visualization tools/techniques. We have seen that patterns are valuable tools, for capturing and communicating the acquired understandings/experience with visualization techniques, to guide in proper selection of visualization tools for solving the visualization problems under consideration.

Like the benefits of using patterns in other works (for example: Georgiakakis et al., 2006), we also believe that use of the visualization pattern approach will help minimize the overhead in the preparatory phase of the evaluation process of visualization techniques, and also allow any novice user to understand the functionality of the visualization technique without little or no assistance. To ease the participants' comprehensibility of visualization techniques that are used in the static software visualizations tools (i.e. SA4J and Creole) under our investigation, we have prepared a simplified version of corresponding patterns (given in 'Appendix C'). These patterns are used by our participants to get an overview of the visualization techniques during the controlled experimentation with these techniques.

Chapter 8. Put It All Together – Comprehension Model for Visualization Assessment (CoMoVA) Framework

"Knowledge comes by taking things apart: analysis. But wisdom comes by putting things together." – John A. Morrison

Overview

This chapter describes our proposed measurement framework referred as ‘CoMoVA’ to measure comprehension support provided by visualization systems, and is based on integrating the knowledge gained from our earlier investigations in previous chapters. In this chapter, firstly we present a clear working definition of what we mean by comprehension and then outline our proposed framework by describing in detail the various components of this framework and the activities to be carried out by an evaluator and participants. Secondly, we provide an example scenario to illustrate the usage of our framework. Finally, we discuss the conformance of the framework to existing measurement models and issues concerning its overall validation.

8.1 Comprehension: A Working Definition

Whenever, we talk of comprehension assessment we cannot perform it independently of the notion of ‘intent’, which can be the intent of the visualization system or of the visual itself or of the user who uses the visualization system. Every visualization system will have its own intention:

- usually to provide easier comprehension of some aspects of the data by interacting suitably with visual representations of the data, e.g. trends through a graph display, and/or
- to enable the user to comprehend ‘hidden’ aspects of the data, say, unknown associations/relationships in data through other visual forms.

Comprehension measurement should address how well the visualization system's intent is met through its visuals and interaction techniques, and how well the user's intent is met by the system.

Moreover, comprehension performance always depends on ‘context of use’ that includes – users’ profiles (i.e. who the users are), tasks’ characteristics, and hardware, software, physical or organizational environments. Lack of knowledge about context, in which the visualization tool/technique is used, may lead to unrealistic comprehension measurement plan. A detailed description of the elements of ‘context of use’ is given below.

- **Users and their characteristics**

We know that users are not a homogeneous group of people and they differ from each other in many ways as follows –

a) Physical factors

It includes factors like – age, gender, vision, and spatial-ability (left-handed or right-handed). These are explained as follows.

Age: Age is a factor that is considered in almost all the empirical studies, as it affects directly the performance of an individual involved in study. We human beings have limited amount of working or short-term memory that reduces with our age, and we lose our ability to recognize and remember the things we used to remember when we were young and had good memories. That is our ability to comprehend visual information is affected by our age.

Gender: Gender is found to be a good predictor of navigation performance, with males outperforming females (Velez et al., 2005). A number of studies (for example – Cutmore et al., 2000; Hubona and Shirah, 2004 and so on) emphasize the fact that gender differences have a strong effect in virtual reality navigation.

Vision: The readability in visualizations is also affected by the capacity of human's eye to perceive various perceptual attributes like – color, shape, lines etc. Color is a basic perceptual attribute that is extensively used in most of the visualizations. It has been observed that 8% of the male population is colorblind against only 2% of the female population.

Spatial-ability: Spatial-ability is regarded as the skills involving retrieval, retention and transformation of visual information in a spatial context (Halpern, 2000). A study by Velez et al. (2005) reveals that spatial-ability is related to visualization comprehension and individuals have highly variant spatial-abilities.

b) Socio-cultural factors

It consists of background and education, which are described as under.

Background: As stated earlier users are not always a homogeneous group of people. They come from different cultures and have different first languages. They may have culturally different meanings for the same terms or icons used in visualizations.

Education: In almost all empirical evaluations, education level is listed as one of the basic characteristic to classify the users.

c) Knowledge and Experience

It comprises application domain knowledge, expertise, programming language knowledge, and the familiarity with the software products under study. These are described as under.

Application domain knowledge and expertise: An acquaintance of the users with various domain concepts helps them to comprehend the underlying information. For example, experience with various software maintenance activities helps the users to look for specific tasks in software visualization tools.

Domain specific skills: For example, in case of software visualization systems, hands on experience with the source language of the software that is visualized should also be considered. A user who knows the programming language very well can comprehend the structure of the visualized software system quite easily.

Familiarity with the visualization tools: A-priori knowledge and experience of the users with the visualization tool also counts when they try to infer information from the tools that they are already familiar with. Such users can more easily perform different tasks with the visualization tools under study.

- **Tasks**

It includes the type of task, complexity of the task, time to perform the task, cost constraints and other task related factors. Generic user tasks for any visualization system are –

Search: The users search in visualization systems for specific items or look for patterns in displayed visuals.

Browse: The users browse the visual space in order to explore it.

Analysis: The users perform suitable analysis operations to make comparisons, seek differences, and to find outliers or extreme patterns.

Assimilation: The users attempt to understand and to learn some new concepts from the data being visualized.

Monitor: The users examine some potential events.

Awareness: The users are made aware of some critical conditions.

- **Environment**

It incorporates a number of factors like – hardware platform (e.g. PC, laptop, handheld computer, mouse, keyboard etc.), software platform, noise level, ambient qualities, type of references and access to experts etc. All these environmental factors affect the way in which the user can interact with a visualization system.

Based on above discussion, we do not see comprehension in isolation from ‘context of use’ and therefore define comprehension as –

The degree to which information represented through visualization can be grasped and interpreted correctly in a specified context of use.

8.2 CoMoVA- An Integrated Comprehension Measurement Framework for Visualization Systems

A framework by its definition is a basic conceptual structure used to solve or address complex issues and a ‘conceptual framework’ is used in research to outline possible courses of action or to present a preferred approach to an idea or thought (Wikipedia, 2007). A measurement framework, in general, is a supporting structure where measurement activities can be carried out. It defines a measurement environment where a set of related metrics and data collection mechanisms can be applied to assess the value of interested features.

Our measurement framework (shown in Figure 8.1) is a systematic structure that links various artifacts to deal with the measurement of comprehension in visualization systems, and is derived from integrating a set of concepts that we have learned in previous chapters. The framework termed as ‘Comprehension Model for Visualization Assessment’ (CoMoVA) includes a protocol for controlled experimentation of visualization systems. Below, we provide answers to the basic questions of who, what, when and how for this framework –

- *‘Who can use the framework?’* – the primary stakeholders in this model are an evaluator and the participants of the controlled experiment for evaluation of the visualization systems in hand. In addition, usability experts may reuse the proposed comprehension criteria and measures to specify design rules or heuristics for visualization systems.

- *'What can be done with this framework?'* – the activities that can be performed by the evaluator (illustrated as 1 to 5 in Figure 8.1) and participants (depicted as 1' to 5' in Figure 8.1) for measuring the comprehension support of visualization systems.
- *'When is it appropriate to conduct evaluation/assessment?'* – the artifacts that are required by the evaluator and participants during controlled experimentation are available, so as to enable measurement of the comprehension support of visualization systems. For example – the questionnaires, repository of comprehension criteria, visualization patterns and tasks.
- *'How can we achieve the main objective of assessing comprehension?'* – through the methods and techniques used to propose the set of comprehension criteria, questionnaires and task catalogue. For example, as illustrated in Figure 8.1, the opinions of HCI/Usability experts collected through interviewing technique, existing principles and case studies used to propose the hierarchy of comprehension factors, criteria and measures.

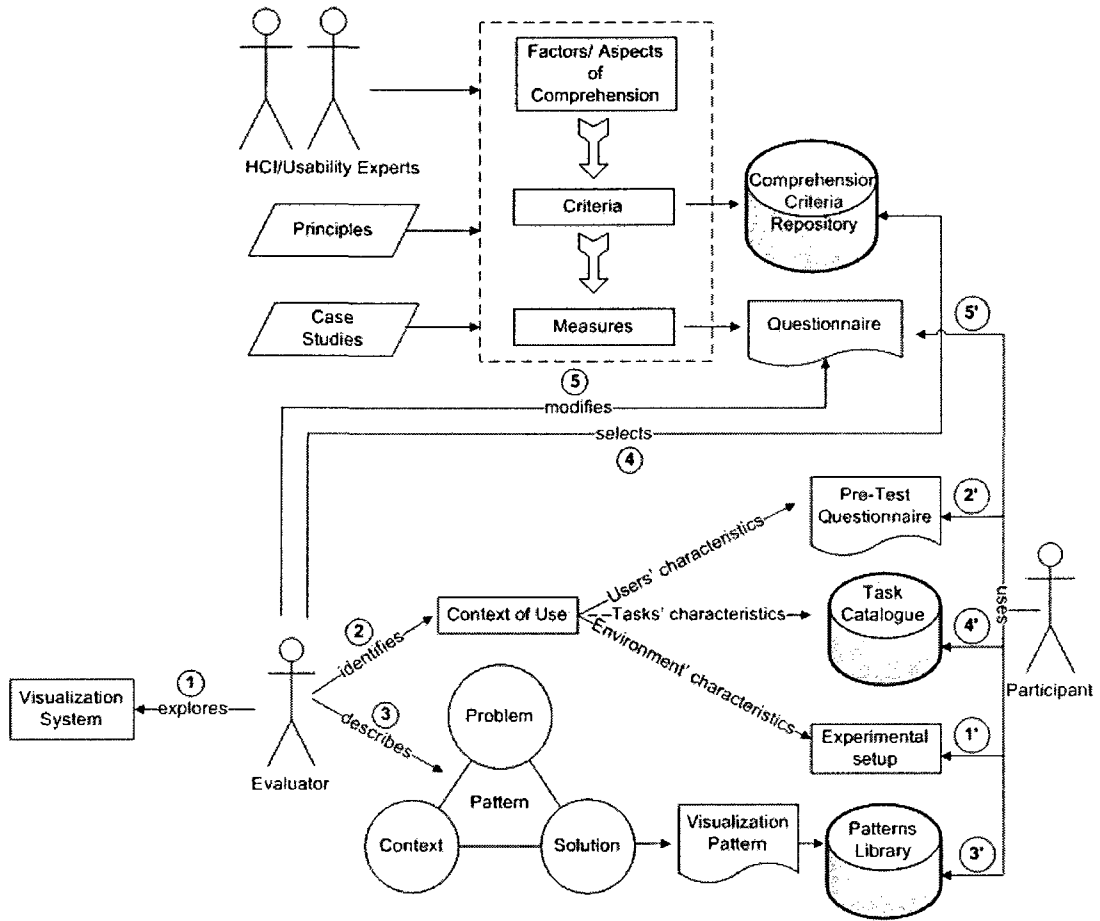


Figure 8.1: The Proposed Comprehension Framework for Visualization Assessment

Like other software engineering models (for example – GQM (Goal Question Metric), ISO 9126, and QUIM (Quality in Use Integrated Measurement)), our measurement model also deals with the measurement of comprehension by characterizing it first in terms of factors or aspects of comprehension. These three factors (i.e. Perception, Presentation, and Cognition) are then sub-divided into 11 measurable criteria as shown in Figure 8.2. Finally, for each of the proposed criteria a number of measures based on answers to questions are derived to measure them. The questions are numbered

(c.f. Figure 8.2) according to their order in the proposed questionnaire given in Appendix ‘A’.

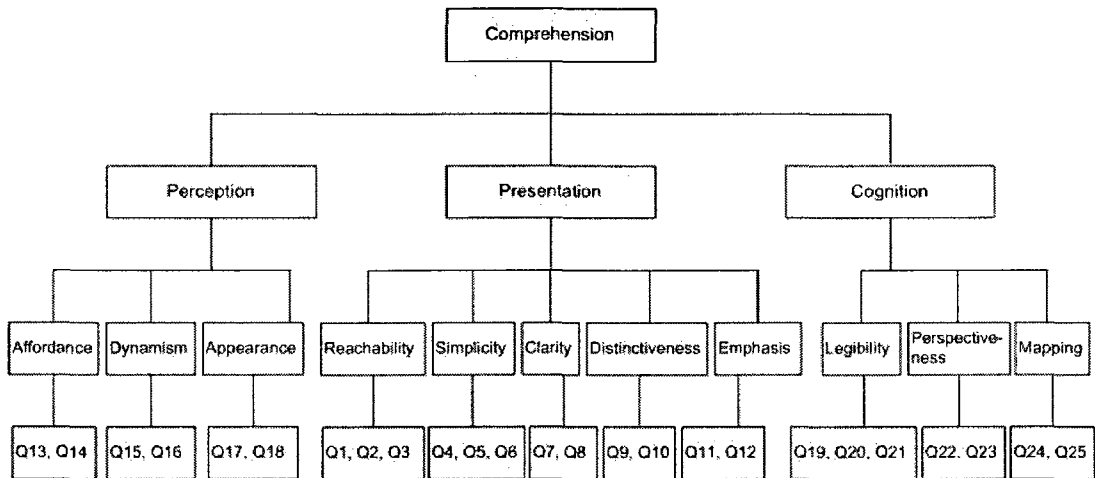


Figure 8.2: The Proposed Comprehension Model for Visualization Assessment (CoMoVA)

This hierarchy of Factors → Criteria → Measures is derived using the inputs from three sources (c.f. Figure 8.1) as follows –

1. HCI/ Usability Experts – We sought the opinions of HCI experts on the perceptual, cognitive and presentation capabilities of visualization systems. Two open-ended interviews were conducted with two experts.
2. Principles – To seek the appropriate criteria to measure comprehension of visualization systems, we sought guidance from two sets of well-established HCI principles. Three visual communication principles proposed by Marcus (Marcus, 1995) i.e. ‘Principle of Organization’, ‘Principle of Economization’ and ‘Principle of Communication’ along with Norman’s cognitive principles (Norman, 1990) such as ‘Affordances’, ‘Mapping’ etc. from the theory of *human action cycle* are our guiding principles. We believe that these basic principles are fundamental for the overall

comprehension of any visualization system regardless of its domain. These guiding principles are applied to determine their affect on various aspects of human comprehension and to derive the corresponding comprehension criteria.

3. Case Studies – Two case studies with two different visualization systems have been conducted to further verify the proposed criteria. These case studies conducted in different domains help us test our proposed hierarchy of comprehension factors, criteria and measures.

Each element of the proposed hierarchy is explained in detail as follows -

□ Factors

It represents three aspects of comprehension i.e. – ‘Visualization Interface’ or ‘Presentation’, ‘Perception’ and ‘Cognition’ as studied in chapter 2 of this thesis. The fourth comprehension aspect i.e. ‘Information Structure’ lies outside the scope of this research. The ‘Visualization Interface’ or ‘Presentation’ aspect represents the presentation capabilities of a visualization system i.e. those visual characteristics that ease the comprehension of underlying information, whereas the ‘Perception’ and ‘Cognition’ aspects signify those properties in a visualization system that ease the users’ visual and cognitive abilities to perform certain other functions with it. These aspects are interrelated as they affect each other for the overall comprehension process. These three aspects or high-level factors are further mapped into measurable criteria as discussed further.

□ Criteria

A total of 11 criteria (i.e. Reachability, Simplicity, Clarity, Distinctiveness, Emphasis, Affordance, Dynamism, Appearance, Legibility, Perspective-ness, and Mapping) have

been derived for these three aspects of comprehension with guidance from the well-established and recognized principles in HCI community. To measure these criteria, a number of questions have been proposed in the next stage.

□ Measures

We have devised a sample questionnaire (given in Appendix ‘A’), comprising a set of questions for each criterion, to be asked in a controlled experiment. The questions address those features of the visualization systems that have impact on corresponding comprehension criteria. For example, one of the questions to measure *Simplicity* criteria is as follows:

- Does the organization of menus seem logical (i.e. are related tasks grouped together)?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

The questionnaire is designed using a three-point (i.e. ‘Yes’, ‘Somewhat’, and ‘No’) rating scale, where a detailed explanation is asked for each middle (i.e. ‘Somewhat’) answer. The subjective response from the participants is then statistically analyzed to compute the total comprehension score of each individual participant.

8.2.1 Activities in CoMoVA

Our CoMoVA framework describes a variety of tasks and activities that take place during the process of comprehension measurement of any visualization system. These activities/tasks are illustrated from the viewpoint of two main stakeholders i.e. – an evaluator and a participant, involved in the controlled experimentation of these systems.

Evaluator Activities

Before the actual evaluation of a visualization system by the participant, an evaluator has to perform the following main activities (shown as 1 to 5 in Figure 8.1) during the preliminary phase of the evaluation.

1. An evaluator begins with the exploration of the visualization system under study to test any difficulties or problems related to its' running, i.e. verifies if the system is utilizable or not.
2. He/she then identifies the 'context of use' of the visualization system. The 'context of use' is a basic requirement to begin any evaluation, as it captures the boundaries of evaluation. The evaluation environment should be described clearly in terms of users' characteristics, tasks' characteristics and environment's characteristics, so that the elements that may influence the evaluation are appropriately summed up. Each of these elements of the 'context of use' contributes to various artifacts in our CoMoVA framework as follows.

Pre-test Questionnaire – The study of users' characteristics enables the evaluator to better understand the target participants from the user population. The users/participants are screened using a pre-test questionnaire that comprises a set of questions related to their physical factors, background knowledge and expertise. An example of a pre-test questionnaire used for static software visualization systems under our study is illustrated in Appendix 'D'.

Task Catalogue – Tasks are a key component of any empirical investigation, as they enable an evaluator or a user/participant to understand the functionalities of a visualization system. An evaluator needs to set up a catalogue of tasks that are

supported by the visualization system. He/she can then select appropriate tasks from this set that are to be asked to the participants during experimentation.

Experimental Setup – The environment characteristics of a visualization system capture the hardware, software, physical and social context of its applicability in the real world. To simulate the same environment during the experimentation of the visualization system with the participants, an evaluator has to clearly identify this element of ‘the context of use’ thoroughly. The study of environmental characteristics leads to a proper experimental setup of the corresponding visualization system.

3. As we have seen in chapter 7, a visualization tool/technique can be good in one context and bad in another. So, for accurate comparisons of visualization techniques, an evaluator needs to encapsulate a visualization technique in a pattern using the applicable context for it. Each visualization technique should be expressed in terms of a visualization problem for which it is suitable, the applicable context and the proposed design solution. Examples of these patterns for software visualization systems are given in Appendix ‘C’ of this thesis. These patterns are stored in a pattern library to assist users/participants during the experimentation of visualization systems.
4. An evaluator has to select appropriate criteria from the proposed repository of comprehension criteria, as some of the proposed criteria may not be suitable for study purposes. For example, while studying the comprehensibility of static software visualization systems under our investigation we did not consider the ‘Dynamism’ criteria from this repository. As the investigated visualization systems were static in nature, this criterion was not applicable.

5. Based on the decision to select appropriate criteria, an evaluator has to modify the proposed questionnaire to be asked to the participants.

Participant Activities

Our CoMoVA framework facilitates the following user/participants tasks (shown as 1' to 5' in Figure 8.1) during the controlled experimentation of visualization systems -

1. A participant uses the experimental setup established by the evaluator.
2. He/she fills in pre-test questionnaire to describe various users' characteristics.
3. He/she then uses the description document from the visualization pattern library to understand the problem the visualization technique is addressing, the context in which it is used and the proposed design solution.
4. After reading the pattern description, the participant is asked to perform a set of assigned tasks from the task catalogue.
5. On performing the required tasks with the visualization system in hand, the participant is asked a questionnaire to evaluate its' comprehension support as perceived by him/her for various comprehension criteria.

8.3 How to Use the Framework?

In this section, we are giving an example scenario to demonstrate how our framework can be applied for evaluation of visualization systems.

Usage scenario for comparative evaluation

Assume a person comes with two visualization systems and the task lists to be performed using these systems. He wants to know which system is more effective in terms of comprehension of the underlying information for accomplishing these tasks.

Solution

An evaluator using our CoMoVA framework will achieve this goal by designing and conducting a comparative test. In this comparative test, there are six stages.

1. Firstly, he/she needs to define a specific 'context of use' in which the test will be conducted to compare two visualization systems. The 'context of use' will comprise the test-users who mimic the original users of visualization systems, the task characteristics in terms of the task size and actual allotted time for the tasks, and the environment in terms of hardware and software platform that will be used for these systems. There can be some elements of 'context of use' that are different, for example one visualization system is using Windows environment and other one is using UNIX environment.
2. The evaluator has to focus on a specific visualization problem that the underlying visualization techniques in both the systems can solve; otherwise it is not feasible to compare them.
3. The evaluator has to describe each visualization technique in terms of a visualization pattern. As stated earlier, the visualization problem is same; it is the solution that varies in these two systems. CoMoVA framework provides a template for defining these patterns.
4. Then, the evaluator will select the most appropriate criteria from our set of proposed criteria in CoMoVA. These criteria are the indicators that help in overall assessment of comprehension. An evaluator can select those that he/she wants to focus in his/her evaluation.
5. The evaluator will design a controlled experiment, where a set of selected users will be asked to do specific tasks with two visualization systems using visualization

patterns as user guides. During the test, two type of information will be collected: observation and measure (or data to calculate measure).

6. The last stage will consist of interpretation and analysis of the measurement results. The analyst will compare the results of test based on measures with the results from observation.

In this process of comparative evaluation, the framework provides help in the following tasks.

- Define the context of use
- Define the visualization patterns
- Provide predetermined assessment criteria
- Define measures
- Compare the measure and observation

8.4 Conformance to Measurement Theory

8.4.1 CoMoVA is a Quality Model

According to Firesmith (2003), “a quality model first decomposes the general concept of quality to create a taxonomy of its component quality factors and sub-factors (i.e., aspects, attributes, or characteristics). The quality model then provides specific quality criteria (i.e. descriptions) and measures (i.e. means of measurement) that can be used to turn these general high-level quality factors into detailed and specific measurable descriptions that can be used to specify the associated aspect of quality or to determine during testing if that aspect of quality actually exists.”

In accordance with this definition, our Comprehension Model for Visualization Assessment (CoMoVA) also creates a hierarchy of quality factor (i.e. Comprehension),

sub-factors (i.e. Perception, Presentation and Cognition), criteria and measures as shown in Figure 8.2. Moreover, the same standard approach of dividing the high-level factors into low-level measurable attributes has been applied in many software engineering models (for example – McCall, Boehm, ISO 9126, QUIM and so on). Therefore, we believe that our top-down measurement model conforms to existing standards in measurement theory.

8.4.2 CoMoVA and Its’ Relationship to ISO 9126

ISO 9126 (2001) is the most recent standard that is applied by many software and system-engineering professionals to measure some aspect of the quality of products. We therefore wanted to see the relationship between our proposed CoMoVA and ISO 9126.

As we stated earlier (in chapter 3, section 3.3.1), ‘Quality in Use’ is the combined effect of six software product quality characteristics and is determined in terms of effectiveness, productivity, safety and satisfaction (c.f. Figure 8.3). We can see that ‘Understandability’ is a sub-characteristic of the ‘Usability’ of any software product in this model.

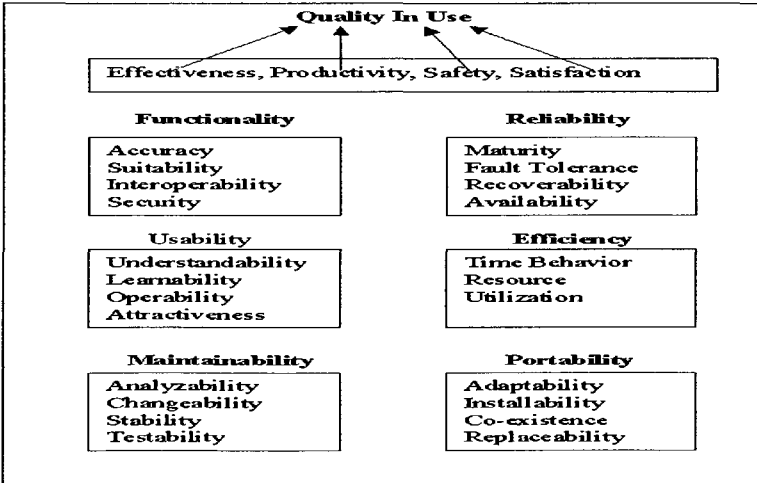


Figure 8.3: ISO 9126

According to Cioch (1991), Understandability consists of two components: ‘Comprehension’ and ‘Lack of Misinterpretation’. Therefore, we believe that our set of comprehension criteria affects the ‘Understandability’ characteristic, which in turn affects the ‘Usability’ quality characteristic, and ultimately affecting the ‘Quality in Use’ characteristic of any software product. Therefore, we can conclude that if a visualization system supports its’ users to comprehend underlying information using the applied interaction mechanisms, then its’ users

- are able to achieve the specified goals with accuracy and completeness in a specified context of use,
- are able to expend appropriate amount of resources in relation to the effectiveness achieved in a specified context of use,
- are aware of the safety issues in a specified context of use, and
- express overall satisfaction with that system.

We will see the effect of comprehensibility of an individual on some of these four high-level quality characteristics in the next chapter of this thesis.

8.4.3 Overall Validation Issues

This work is the first and foremost in the current state of comprehension measurement of visualization systems. Validating the framework is a long-term objective that will involve conduct of large-scale experiments with different visualization systems using different contexts. Such a work goes beyond the scope of this thesis. As a first stage and part of this research, we conducted a controlled experiment with two static software visualization tools to test the applicability and effectiveness of our proposed CoMoVA framework. This experiment is explained in detail in next chapter.

Chapter 9. Operationalization and Overall Validation of the Framework

“There are two possible outcomes: if the result confirms the hypothesis, then you've made a measurement. If the result is contrary to the hypothesis, then you've made a discovery.” – Enrico Fermi (1901-1954)

Overview

In this chapter, we describe in detail a controlled experiment conducted by us to demonstrate the use of our proposed CoMoVA framework. In this experiment, two static software visualization systems, SA4J (Structural Analysis for Java) and Creole are studied for their comprehension support using our CoMoVA framework. A total of 15 participants from the university community were invited to perform a controlled experiment with these visualization systems in our human-centered software engineering lab. The participants were asked to perform various activities outlined in CoMoVA framework using the proposed artifacts (i.e., visualization patterns, questionnaire etc.). The responses from the participants were statistically analyzed to validate their scores.

9.1 Measurement Goal Template

Freimut et al. (2001) suggest five key elements of an experiment and organize them in the form of a measurement goal template as shown in Table 9.1. According to them, the purpose of a measurement goal template is to ensure that important aspects of an experiment are defined before planning and execution take place (Freimut et al., 2001).

Table 9.1: Measurement Goal Template (Freimut et al., 2001)

Elements of an experiment	Definition	Measurement Goal
Object(s)	The entity that is studied or observed in the experiment. It can be product, process, model, metric or theory.	Software visualization tools that visualize the static structure of a software system.
Purpose	It defines the intention behind the experiment. It is closely connected to the research question.	The purpose of the experiment is to determine the effectiveness of visualization tools/techniques to the users in terms of their supported comprehension i.e. to quantify the comprehension performance of these tools objectively.
Quality focus	The quality focus is the primary effect under study in the experiment.	Quality focus of the experiment is the users' comprehension performance with the visualization systems.
Perspective	It tells the viewpoints from which the experiment results are interpreted.	The perspective of this experiment is mainly the user who uses these tools.

Table 9.1 (continued)

Elements of an experiment	Definition	Measurement Goal
Context	The environment in which the experiment is run. The experiment context can be characterized in terms of the characteristics of subjects (or participants) and objects involved in the study, along with the domain in which the experiment is conducted.	Software professionals who have the knowledge and expertise in the field of software maintenance and visualizations in general conduct the experiment. All categories of users (i.e. novice, intermediate and expert software professionals) are the subjects of the study to understand the differences among different users' characteristics and their impact on overall comprehension of visualizations provided by the software visualization tools. The objects or artifacts of the study is a software project coded in object-oriented language mainly Java, with enough size and complexity to be a realistic example of the projects that are encountered by software maintainers in their regular routine work.

The main contributing factor i.e. 'context' in Table 9.1, which affects the empirical performance of an experiment, is explained in detail as follows:

9.1.1 Context

To be repeatable, an experiment has to be conducted under a specific context where each of its elements should be described in detail. In HCI terminology, context of an experiment can be described using three basic dimensions – user characteristics, task characteristics, and environment characteristics in which the experiment is conducted. Each of these plays a critical role in the planning and execution of an experiment and is described in detail in this section.

- **Users and their characteristics**

As we are going to investigate static software visualization tools mainly used for the purpose of maintaining software, it is quite obvious that our users should have some background knowledge of fundamental practices in software maintenance. Based on our previous discussion of various user characteristics (c.f. chapter 8, section 8.1), we have prepared our pre-test questionnaire (c.f. Appendix ‘D’).

- **Tasks**

The tasks selected for evaluation should be representative of what the users do with the visualization systems and must be manageable and suitable for a laboratory evaluation. Ideally speaking, to judge the comprehensibility of visualization systems, the users should be free to explore anywhere in the visualization and should explore almost all the functionality offered in the visualization system. However, this is not feasible within the time constraints of a controlled experiment. Therefore, we adopted an alternative approach where we ask the users to explore freely for first few minutes of their test, and then ask them to perform one simple task, which is regarded as the main task to be performed using the visualization system.

The chosen task

The task we have chosen for evaluation is a concrete task, a core-level program understanding task that involves understanding of a small portion of the test program. Our test program i.e. BORG (Berger-Organizer) calendar system (c.f. chapter 7, section 7.3.3) consists of 54 packages, 172 classes, and 13 interfaces. The visually presented form of this complete system is cluttered and it is hard for any person to understand the relationships among various components of the system. Therefore, for our controlled experiment we used only a small portion of this system and studied the visual affect of 1 package containing 40 classes and 1 interface. This package, the main component of the whole system, performs most of the functionality of the software. The chosen package is also the largest package in terms of its size among all the packages of the calendar system. Specifically, for this particular experiment, the participants were asked to perform a simple search task as follows:

- Find a class “MultiView”, and related information objects (i.e. classes, packages or interfaces) in the visualization.

The goal of this task is to see if the visualization system supports effective graphic layout where it is easy to find the relevant information. A reasonable time limit of 5 minutes is set to ensure that all the participants can complete this task.

▪ Environment

In our case, the environment is kept same for all the visualization tools/techniques used for our study. The hardware and software needed for the experiment are summarized in Table 9.2.

Table 9.2: Installed Hardware and Software

Hardware	Software
<ul style="list-style-type: none">▪ PC - Dell, Intel Pentium (R) 4 CPU 2.80 GHz, 1:00 GB RAM▪ Camera - Logitech's QuickCam Pro 4000▪ Microphone system - Sony's WCS-999 Wireless Microphone System consisting of a wireless transmitter and receiver▪ Keyboard, mouse	<ul style="list-style-type: none">▪ Microsoft Windows XP Professional version 2002,▪ Java Run-time Environment (JRE 1.4.2_13 and 1.5.0_12),▪ Berger Organizer (BORG) Calendar system,▪ Eclipse Software Development Kit (SDK 3.2.2)▪ TechSmith's MORAE usability testing software

9.2 An Exemplar Study – A Controlled Experiment with Software Visualization Tools

This section describes a controlled study to evaluate the comprehension support of four visualization techniques applied in two static software visualization tools. This study was conducted in human-centered software engineering lab at Concordia University in winter 2008.

9.2.1 Goals

We had four main goals in mind as follows.

1. To demonstrate that the framework CoMoVA is usable and can be used.
2. To observe whether the visualization patterns are really useful in assisting the users to understand the underlying visualization techniques.

3. To determine the comprehensibility of four visualization techniques employed in two tools based on the responses of the participants to the corresponding questionnaires.
4. To compare the effectiveness of one visualization tool with another, based on the cumulative comprehension scores of various participants.

9.2.2 Participants

For the experiment, 15 participants (7 females and 8 males) having some experience of using visualization systems in general were recruited from the university community. Prior to the actual experimental sessions, we asked each participant to complete a pre-test questionnaire as given in Appendix 'D' of this thesis. Through this questionnaire, we collected various background variables as shown in Table 9.3 to categorize our participants based on their knowledge and experience in software maintenance domain.

Table 9.3: Background Variables

Variable	Number	%	Mean	SD	Range
Gender					
Female	7	46.67			
Male	8	53.33			
Age (years)			29.20	6.64	23-46
First Language					
English	2	13.33			
Other	13	86.67			
Education level					
Bachelor	0	0.00			
Masters	9	60.00			
PhD	6	40.00			
Left handed					
No	12	80.00			
Yes	3	20.00			
Color-blinded					
No	15	100.0			
Yes	0	0.00			
Number of graduate-level software maintenance courses taken					
None	7	46.67			
1-2 courses	8	53.33			
>2 courses	0	0.00			
Knowledge of software maintenance					
None	0	0.00			
Basic	4	26.67			
Intermediate	10	66.66			
Advanced	1	6.67			
Experience with chosen software visualization tools (i.e. Creole and Structural Analysis for Java (SA4J))					
None	12	80.00			
Basic	2	13.33			
Intermediate	1	6.67			
Advanced	0	0.00			
Experience with Java language					
None	0	0.00			
Basic	2	13.33			
Intermediate	9	60.00			
Advanced	4	26.67			

We further applied a grouping scheme as follows to classify our participants.

Grouping scheme

The participants were grouped in three groupings (i.e. Novice, Intermediate, and Expert) based on their knowledge of software maintenance in general and the number of graduate-level software maintenance courses taken by them. This scheme is described as follows.

Novice – A novice is a subject who has basic knowledge of software maintenance domain and has undertaken at most 1 graduate-level software maintenance course i.e. ‘Basic’ \wedge (≤ 1 course) \rightarrow Novice

Intermediate – An intermediate is a subject who has ‘intermediate’ knowledge of software maintenance domain and has completed ‘0’ graduate-level software maintenance course, i.e. ‘Intermediate’ \wedge (no courses) \rightarrow Intermediate

Expert – An expert is a subject who has advanced or intermediate knowledge of software maintenance domain and has completed at least ‘1’ graduate-level software maintenance course, i.e. (‘Advanced’ \vee ‘Intermediate’) \wedge (> 0 courses) \rightarrow Expert

9.2.3 Hypothesis

As already stated earlier in chapter 7, each visualization system employs a number of visualization techniques to facilitate comprehension of the underlying information and therefore, the assessment of comprehension support needs to be conducted for the corresponding visualization techniques. In our study, we are investigating the comprehensibility of two static software visualization tools – SA4J (Structural Analysis for Java) and Creole. SA4J employs two visualization techniques termed as ‘Radial’ and ‘Pyramid/Skeleton View’ to depict the static structure showing dependencies in a

software system. On the other hand, Creole uses five different visualization techniques to display the static structure of a software system. Out of these five visualization techniques, only two techniques named as ‘NestedView’ and ‘Tree’ are explored within the timeframes of our controlled experiment. Therefore, for our experiment we invited participants to explore and comment on the comprehensibility of four visualization techniques i.e. Radial, Pyramid/Skeleton, NestedView, and Tree.

To assess the comprehension support of these visualization techniques, we outlined a null hypothesis of this experiment as follows.

Null hypothesis – *Radial, Pyramid, NestedView and Tree techniques are equally effective in terms of comprehension under the same conditions.*

In order to validate our results obtained from the proposed questionnaire to assess comprehension, we further studied the users’ task performance for a simple exploratory task. The chosen task is supported by all the underlying visualization techniques. We captured this additional effort in a hypothesis as follows:

H1 – *The users’ task performance with visual representations should depend on the comprehension support of underlying visualization techniques as assessed by our questionnaire.*

9.2.4 Experimental Variables

The independent variables in the experiment are:

- the visualization tools (i.e. SA4J and Creole) and the underlying visualization techniques (i.e. Radial, Pyramid/Skeleton, NestedView, and Tree),
- the software program (i.e. Berger-Organizer) that is visualized,
- knowledge and expertise of the participants, and

- the complexity of software maintenance task.

The following dependent variables are assumed to be influenced:

- the comprehension score as assessed for various criteria like - Reachability, Simplicity, Clarity, Distinctiveness and so on,
- time taken to complete the assigned task, and
- correctness of the performed task.

9.2.5 Types of Experimental Biases and Their Elimination

Many practical difficulties may arise in running an empirical experiment. Although, we cannot entirely prevent experimental biases but we can enumerate them, as shown in Table 9.4, to minimize their overall affect on our results.

Table 9.4: Classification of Experimental Biases

Cause	Affect	Remedy
Single experimenter	Although, one experimenter will reduce the communication difference that may arise with several experimenters. However, one can feel tired or bored by repeating the same information to the users or participants, and as a result can miss some pertinent information to the test.	There should be a significant gap between two consecutive tests and experimenter should consult an experimenter's handbook while conducting a test.
A fixed order of studying each tool	It is quite natural that users may get tired of executing the same task for each tool and it may impact their comprehension performance of the later tools being studied.	Introducing short breaks within each tool study to start afresh for next one. To remove the impact of tiredness on study results, randomize the order in which the tools should be tested by the set of participants.

Table 9.4 (continued)

Cause	Affect	Remedy
A single source project is visualized with many visualization tools.	The knowledge gained by the participant while exploring the source project using one visualization tool can impact on using the same source project in another visualization tool.	Ideally speaking, a different source project of similar size and complexity is needed for each visualization tool. However, our main objective is to measure the comprehension support of different visualization systems for the same underlying software program. Therefore, we left this bias as such and only observed its affect on our participants during the experiment.

9.2.6 Experimental Setup

In any experiment, a well-designed setup is needed to obtain results with reasonable confidence. Towards this objective, we designed various structural elements of an experiment as follows.

9.2.6.1 Experimental Phases

90 minutes to 2 hours session with each of the participants contained two different types of phases as follows.

One-timed: These phases are to be completed only once for each participant and take in total 5 minutes of the total session time. These are – orientation (3 min) and background evaluation (2 min).

Repetitive: These phases should be repeated for each of the visualization techniques. It includes various phases like – training task (2 to 3 min), free exploration (5 min), formal task (maximum of 5 minutes), and questionnaire evaluation (~16 minutes).

A brief explanation of each of these phases is as under.

a) Orientation

The experimenter begins the experiment by welcoming and briefly orienting the participant. Each participant is reminded of the purpose of the experiment i.e. to devise a comprehension model for evaluation of visualization systems. A consent form (shown in Appendix ‘E’) was already electronically mailed to the participants when they were invited for the study. The same consent form is given to the participant to outline the procedure of the study. The participant is informed that the test session will be audio and video recorded for study purposes, and he/she is assured that the collected information will remain anonymous. Also, to relax our participants, it is emphasized that it is the visualization systems and not the participants that are being tested in the experiment.

b) Background evaluation

A preliminary participant evaluation form given in Appendix ‘D’ is given to the participant to determine various background variables for study purposes.

c) Training tasks

To facilitate understanding of the visualization techniques and to ease the preliminary phase of the experiment, the participants are given visualization patterns as a quick user guide. These patterns, as described in Appendix ‘C’, emphasize the functionalities of the corresponding visualization techniques in an abstract manner.

The visualization patterns enable the participant to quickly learn and easily understand the basic features of corresponding visualization techniques.

d) Free exploration

The participants are then allowed to freely explore the tested visualization technique to understand and assimilate the basic concepts displayed on screen. This phase helps the participant to become familiar with the techniques and the supported interaction mechanisms. During this initial exploration, they are instructed to explore anywhere on the display (with an exception for Creole visualization techniques – where they are asked not to explore the upper toolbar belonging to Eclipse platform). The participants are encouraged to ask questions about the visualization techniques.

e) Formal task

After the free exploration, the participants are asked one formal task to search for a specific class in the hierarchy of other information objects (i.e. packages, classes, and interfaces). The participants are also encouraged to think-aloud while performing the assigned task in order to note down the comprehension difficulties encountered by them in the corresponding visualization techniques.

f) Questionnaire evaluation

Participants are asked one questionnaire for each visualization technique to assess the effectiveness of these techniques in terms of comprehension. As both the tools used for evaluation were static i.e. they didn't show the dynamic aspects of the software system, therefore the *Dynamism* criteria in our model did not apply in this case. Thus, the questionnaire presented to all the participants was slightly modified from its' original version as given in Appendix 'A' by excluding the questions to assess

Dynamism criteria. The questionnaire is presented to a participant after the formal task is completed with a given visualization technique. The participants are persuaded to interact with the visualization tools while answering the questionnaires. We selected a Likert-scale having three values as answers to each of the question; where ‘Yes’ means 100%, ‘Somewhat’ is 50% and ‘No’ is assigned 0% value. For any ‘somewhat’ answer, the participants are asked to explain in detail of their reasoning. The ordering of all questions in the questionnaire is kept same for all the participants. The questions are classified to assess ten comprehension criteria as follows:

Reachability: questions 1-3 measure the reachability or navigability in a visualization technique.

Simplicity: questions 4-6 assess the simplicity features of a visualization technique

Clarity: questions 7-8 measure the clarity criteria

Distinctiveness: questions 9-10 evaluate the distinctiveness characteristics of a visualization technique

Emphasis: questions 11-12 enable assessment of emphasis property

Affordance: questions 13-14 corresponds to affordance criteria

Appearance: questions 15-16 deals with the appearance criteria

Legibility: questions 17-19 assess the legibility requirements of a visualization technique

Perspective-ness: questions 20-21 measure the perspective-ness issue of a visualization technique

Mapping: questions 22-23 determine the mapping criteria for comprehension in a visualization technique

In addition to the questionnaire, the following question is asked in the study after a participant completes testing all of the visualization techniques.

1. Rank the four visualization techniques in the order of your likeability to depict the static structure of underlying software system.

9.2.6.2 Dry Run

To discover problems with the initial experimental design, a dry run of the experiment was performed. A dry run of the study was conducted with a novice participant to determine the maximum time-limit for each phase of the experiment. In this preliminary test, we noted down the mistakes in the conduct of the experiment. For example –

Earlier, we were asking a formal task to find an interface called ‘Navigator’ and related information objects rather than a class ‘MultiView’ and related information objects. During the pilot test, we realized that our pilot participant was able to find the asked interface in a very short span of time. This was because there was only one interface in the tested package and almost all the visualization techniques (except Pyramid/Skeleton View) were emphasizing the interface with some visual attribute. Therefore, we realized that with this task we would not be able to compare our results. Thus, we changed our formal task to find a class and related information objects. This class was not readily visible and thus it took some time for the participants to explore the visualizations. In addition, we also observed that some default parameters should be kept constant for all the participants, otherwise our results might vary for each participant under different conditions.

9.2.6.3 Experimenter's Handbook

A detailed experimenter's handbook, as given in Appendix 'F', was written for the experimenter to provide consistency and control over the running of each experimental session. This handbook outlined various general instructions and the checklists (common to all tools) that should be taken into consideration before, during and after each participant's session. It enabled the experimenter to draw the structure of the experiment by following various rules of conduct or procedures to be followed for each successful trial, and provides general instructions on setting up the workstation for the trial. The same copy of the handbook was used for each session. These protocols ensure that the experiment proceeds smoothly and in a consistent manner, reducing the likelihood of mistakes that might affect the results of the study later on.

9.2.6.4 Experimental Procedure

A protocol for the study with each participant is outlined in Figure 9.1. Tests were run one at a time in order to observe the participants using 'Morae' recorder tool. In the study, each test lasted between 1.5 and 2 hours. The experimenter's handbook was used throughout the experiment.

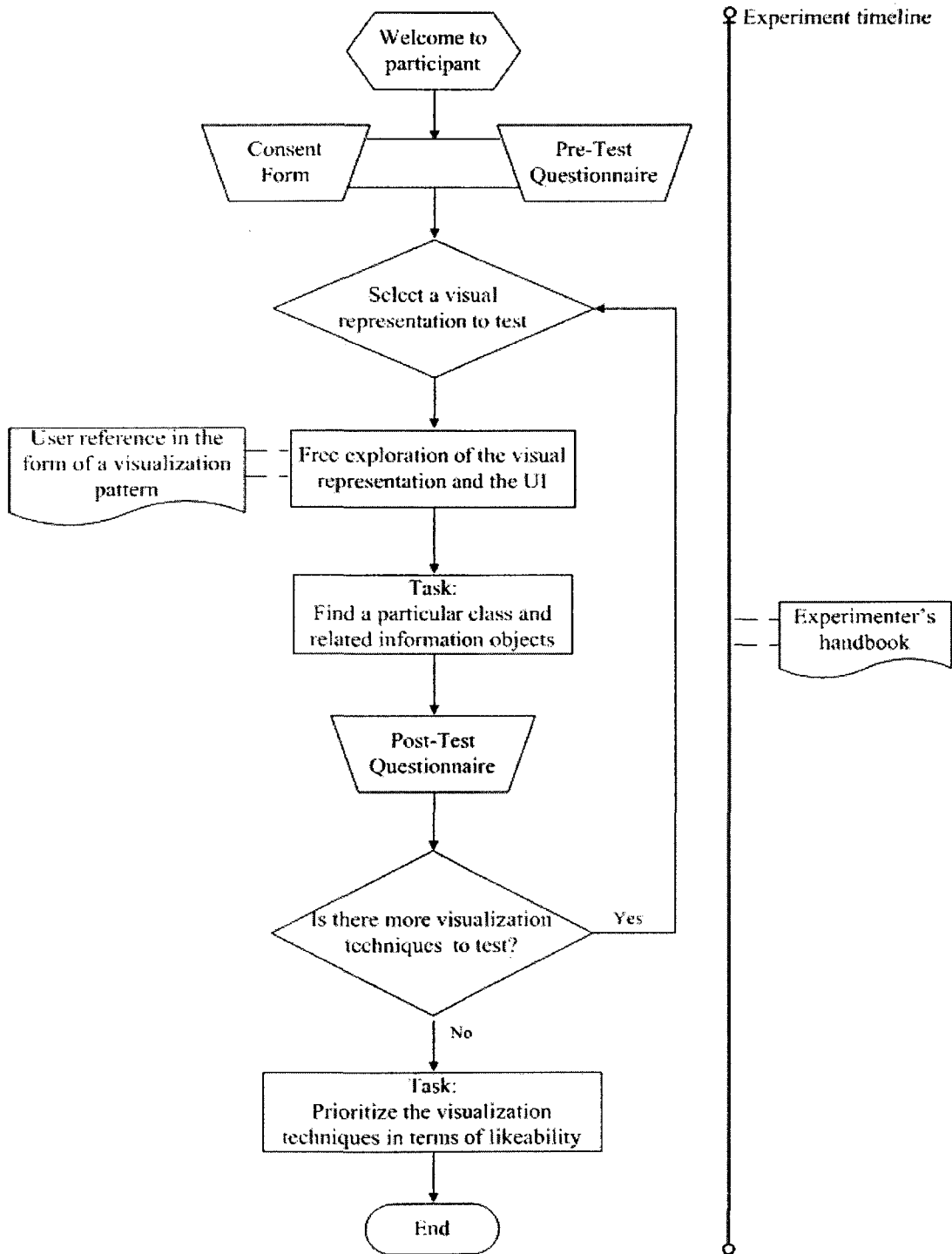


Figure 9.1: Phases of The Experiment

9.2.6.5 Recording Observations

It is not possible to collect all the relevant information from the answers to the questionnaires alone. An experimenter always needs some complementary resources in the form of audio and video recorders that adds to this collected information. These recorded observations can be subsequently used to determine the difficulties experienced by the participants during the test session. In our study, we used several methods of recording observations:

a) Thinking-aloud

The participants were asked to verbalize their thoughts as they explored the visualization techniques and performed the assigned task. This allowed the experimenter to gain a better understanding of what each participant was trying to accomplish during the experiment.

b) Video and audio taping

The video and audio recordings of the test session were captured using the QuickCam camera and the wireless microphone system respectively. Using a Morae Recorder software tool, we were also able to record the user's facial expressions and their actions on the computer screen using the mouse and keyboard.

c) Experimenter comments

The observed behaviour of the participant during the experiment was also written by an experimenter in the form of brief comments.

9.3 Analyzing the Results of a Controlled Experiment

A detailed 'pie-chart' summary of the participants' responses to the questionnaires for each technique is given in Appendix 'G'. Furthermore, we also observed the normal

distribution of the collected responses as shown in Appendix ‘G’. Considering the distribution of participants’ responses normally distributed, the median is same as the mean for that sample of participants. Therefore, instead of calculating median value as the score for each criterion, we devised a measurement strategy to determine mean value as given below.

9.3.1 Measurement Strategy

Similar to the work of (Stavrinoudis et al., 2005), comprehension of a visualization system, from a single user viewpoint, can be expressed in quantitative terms using a weighted arithmetic average of all the criteria as follows:

$$U_i = \frac{\sum_{k=1}^{10} (\text{weight of criterion } C_k * \text{value of criterion } C_k)}{\sum_{k=1}^{10} \text{weight of criterion } C_k} \quad (1)$$

Here,

U_i is the comprehension score of a single user ‘i’;

$C_k = \{\text{‘Reachability’}, \text{‘Simplicity’}, \text{‘Clarity’}, \text{‘Distinctiveness’}, \text{‘Emphasis’}, \text{‘Affordance’}, \text{‘Appearance’}, \text{‘Legibility’}, \text{‘Perspective-ness’}, \text{‘Mapping’}\}$, such that $C_1 = \text{‘Reachability’}$, $C_2 = \text{‘Simplicity’}$ and so on;

k is an integer value in the range [1..10], where number ‘10’ indicates the total number of studied criteria excluding ‘Dynamism’ criterion;

In Equation (1), weight of criterion depicts the relative importance of each criterion to the total comprehension. This relative importance can be derived on a ratio scale by the mutual comparison of all the criteria for comprehension. The weight can be assigned by counting the number of relationships a criteria have with other criteria. For our analysis,

we assume each criterion to be equally important for comprehension, i.e. weight of each criterion is assigned unity or value '1'. This results in the following equation.

$$U_i = \left(\sum_{k=1}^{10} \text{value of criterion } C_k \right) / \text{Total number of criteria} \quad (2)$$

The value of each criterion can be further described in terms of the weighted arithmetic average of the associated measures as follows:

$$\text{Value of criterion } C_k = \frac{\sum_{n=1}^N (\text{weight of measure } M_n * \text{value of measure } M_n)}{\sum_{n=1}^N \text{weight of measure } M_n} \quad (3)$$

Here,

M_n is the related measure for criterion C_k ;

N is the total number of measures that are used to measure corresponding criterion C_k ;

value of measure = $\{x \mid x \in (100\%, 50\%, 0\%), \text{ where } 100\% \text{ means 'Yes', } 50\% \text{ means 'Somewhat', and } 0\% \text{ means 'No' }\}$; and

n is an integer in the range $[1.. N]$

In the same manner as for criteria, the weight of each measure depicts the relative importance of each measure in assessing the corresponding criterion. For making our analysis simpler, we assign equal weight to all the measures associated with each criterion. Therefore, Equation (3) reduces to Equation (4) as follows:

$$\text{Value of criterion } C_k = \left(\sum_{n=1}^N \text{value of measure } M_n \right) / N \quad (4)$$

On combining Equation (2) and (4), we have a combined formula to measure a user's comprehension based on our criteria as follows:

$$U_i = \left(\sum_{k=1}^{10} \left(\sum_{n=1}^N \text{value of measure } M_n / N \right) \right) / \text{Total number of criteria} \quad (5)$$

Finally, in order to measure average users' opinion of the comprehension support of a visualization system, we need to compute the average over the scores of all the users that is weighted by their expertise as follows

$$\text{Total comprehension of a visualization system} = \left(\sum_{i=1}^m Q_i * U_i \right) / \left(\sum_{i=1}^m Q_i \right)$$

Here,

Q_i is the expertise value assigned to a particular user 'i', where set $Q_i = \{1, 2, 3 \mid 1 \equiv \text{Novice}, 2 \equiv \text{Intermediate and } 3 \equiv \text{Expert}\}$;

m is the total number of users/participants that participated in the experiment;

i is an integer in the range $[1..m]$;

With the above formula to measure comprehension of a visualization system, we weigh users' opinion according to their expertise.

Tables 9.5 to 9.8 show the results of each individual participant's score of comprehension criteria for the four visualization techniques. The participants' answers for each of the technique are analyzed using the above formulae.

For example –

Suppose a participant has checked 'Yes' in question 1, 'Somewhat' in question 2, and 'No' for question 3. Then, his/her score for *Reachability* criteria is $(100+50+0)/3$ or 50%. The total comprehension score for each individual participant is the average value of the scores for 10 criteria.

Table 9.5: Participants' Scores of Comprehension Criteria for Radial Technique

Participant #	Criteria	Reachability	Simplicity	Clarity	Distinctiveness	Emphasis	Affordance	Appearance	Legibility	Perspective-ness	Mapping	Total Comprehension
1		100	100	75	100	100	75	75	100	50	100	87.5
2		83.3	100	100	100	100	25	75	100	50	100	83.3
3		100	100	100	75	100	75	25	100	50	75	80.0
4		100	100	75	75	100	75	50	100	50	100	82.5
5		100	66.7	50	75	100	75	50	100	50	75	74.17
6		100	100	100	75	100	100	50	100	50	75	85.0
7		100	83.3	75	75	100	75	75	100	50	100	83.33
8		83.3	100	75	75	100	75	50	83.3	100	100	84.16
9		100	100	75	75	100	100	100	100	100	100	95.00
10		83.3	66.7	25	75	100	100	25	83.3	50	75	68.33
11		100	100	100	75	100	75	50	100	50	100	85.00
12		100	100	100	100	100	100	75	100	75	100	95.00
13		66.7	50	50	25	100	50	50	50	50	75	56.67
14		83.3	83.3	75	75	100	25	75	50	50	75	69.16
15		100	83.3	75	25	100	75	25	83.3	75	75	71.67

Table 9.6: Participants' Scores of Comprehension Criteria for Pyramid Technique

Participant #	Criteria	Reachability	Simplicity	Clarity	Distinctiveness	Emphasis	Affordance	Appearance	Legibility	Perspective-ness	Mapping	Total Comprehension
1		33.3	83.3	50	75	100	75	0	100	0	50	56.67
2		33.3	83.3	50	75	100	100	50	100	0	50	64.16
3		33.3	66.7	50	25	100	50	0	100	25	50	50.00
4		100	83.3	75	50	75	75	0	83.3	25	75	64.16
5		33.3	66.7	75	50	100	75	50	100	0	75	62.50
6		33.3	66.7	100	75	75	100	25	100	0	25	60.0
7		66.7	50	75	50	50	50	0	100	0	50	49.17
8		83.3	100	75	75	100	75	0	83.3	0	50	64.16
9		83.3	66.7	50	0	50	25	50	83.3	0	25	43.33
10		50	66.7	25	0	100	100	25	83.3	0	50	50.0
11		100	83.3	100	75	100	100	75	83.3	50	50	81.67
12		66.7	66.7	50	75	50	75	75	100	0	50	60.84
13		33.3	33.3	25	25	50	75	25	50	0	50	36.67
14		50	33.3	50	50	50	25	50	100	0	50	45.83
15		33.3	50	100	25	100	50	0	100	50	50	55.83

Table 9.7: Participants' Scores of Comprehension Criteria for NestedView Technique

Participant#	Criteria	Reachability	Simplicity	Clarity	Distinctiveness	Emphasis	Affordance	Appearance	Legibility	Perspective-ness	Mapping	Total Comprehension
1		66.7	100	25	25	100	100	50	66.7	0	50	58.34
2		66.7	66.7	75	100	100	50	25	83.3	25	100	69.17
3		66.7	83.3	100	100	100	50	25	83.3	50	50	70.83
4		66.7	100	75	100	100	100	50	100	25	100	81.67
5		83.3	100	50	75	50	50	50	100	0	50	60.83
6		66.7	100	75	100	100	50	50	100	25	25	69.17
7		100	66.7	100	100	100	0	25	66.7	0	100	65.84
8		83.3	100	75	75	100	75	75	100	50	75	80.83
9		50	100	75	100	50	100	25	100	75	75	75.00
10		66.7	50	75	75	75	75	50	66.7	50	75	65.84
11		66.7	100	100	100	50	75	75	100	50	100	81.67
12		83.3	100	75	75	50	50	25	100	25	50	63.33
13		50	66.7	25	25	100	50	50	66.7	0	100	53.34
14		33.3	100	100	100	75	75	50	83.3	75	100	79.16
15		83.3	83.3	75	75	100	100	50	33.3	75	50	72.49

Table 9.8: Participants' Scores of Comprehension Criteria for Tree Technique

Participant #	Criteria	Reachability	Simplicity	Clarity	Distinctiveness	Emphasis	Affordance	Appearance	Legibility	Perspective-ness	Mapping	Total Comprehension
1		66.7	100	75	100	100	75	100	66.7	100	75	85.84
2		50	83.3	75	100	75	100	50	83.3	25	100	74.16
3		50	83.3	100	100	75	50	50	66.7	50	50	67.50
4		50	100	50	100	100	75	75	100	25	100	77.50
5		33.3	66.7	100	75	100	75	0	66.7	0	50	56.67
6		66.7	100	100	100	100	25	50	83.3	0	100	72.50
7		66.7	66.7	50	100	50	100	100	66.7	0	100	70.01
8		83.3	100	75	100	100	75	50	50	100	75	80.83
9		50	50	50	100	100	25	50	83.3	0	75	58.33
10		66.7	50	25	75	75	100	100	50	50	100	69.17
11		66.7	100	50	100	50	75	100	66.7	75	100	78.34
12		83.3	100	50	75	100	75	100	100	75	100	85.83
13		66.7	66.7	25	50	100	50	75	66.7	50	100	65.01
14		0	66.7	50	100	100	75	50	66.7	0	100	60.84
15		100	83.3	50	50	100	50	50	83.3	50	100	71.67

9.3.2 Confirming the Expertise

By applying the grouping scheme as explained in section 9.2.2, we determined that out of 15 invited participants, 6 participants were experts, 5 were intermediates, and 4 were novices in the domain. The opinions of the participants as expressed in Tables 9.5 to 9.8 should be weighted according to their expertise. However, before computing the average participants' opinion on the comprehensibility of respective techniques for this sample of participants we should make sure if our groups of participants (i.e. Novices,

Intermediates, and Experts) came from different populations or not. This can be done by performing an ANOVA (Analysis of Variance) test, which can tell us the non-normality of the groups. Our single factor ANOVA test for Radial, Pyramid, NestedView, and Tree technique is presented in detail in Appendix 'H'.

From our ANOVA results, we have seen that variations within group are higher than variations between groups for all the four visualization techniques. Therefore, based on ANOVA results, we can conclude that there is no significant difference among the opinions of different groups. This means all the three groups (i.e. Novices, Intermediates, and Experts) came from same population; and so, the opinions of all participants are assigned equal weight-age in computing the average comprehension of the respective visualization techniques.

9.3.3 Analysis of the Gender Differences

In our sample of 7 females and 8 males, one-way ANOVA test was computed to see the difference in the means of these two groups. Our ANOVA result (Table H.6 in Appendix 'H') has shown that for all the visualization techniques the value of $F(1, 13) < 1$. This means that there is no significant difference between the scores obtained by males' and females', i.e., the means of these groups are not reliably different.

9.3.4 Validating the Results with Objective Metrics

Furthermore, to validate our results obtained from the participants' responses, we looked at their task performance for the formal task during the experiment. To analyze participants' task performance in terms of task time, and effort in terms of number of mouse clicks, we took the help of Morae manager software. The screen shot of this tool is shown in Figure 9.2. With this tool, we can observe all the recorded events in each test

session and we can also automatically generate various metrics like – task time, number of mouse clicks during a specific period of time etc.

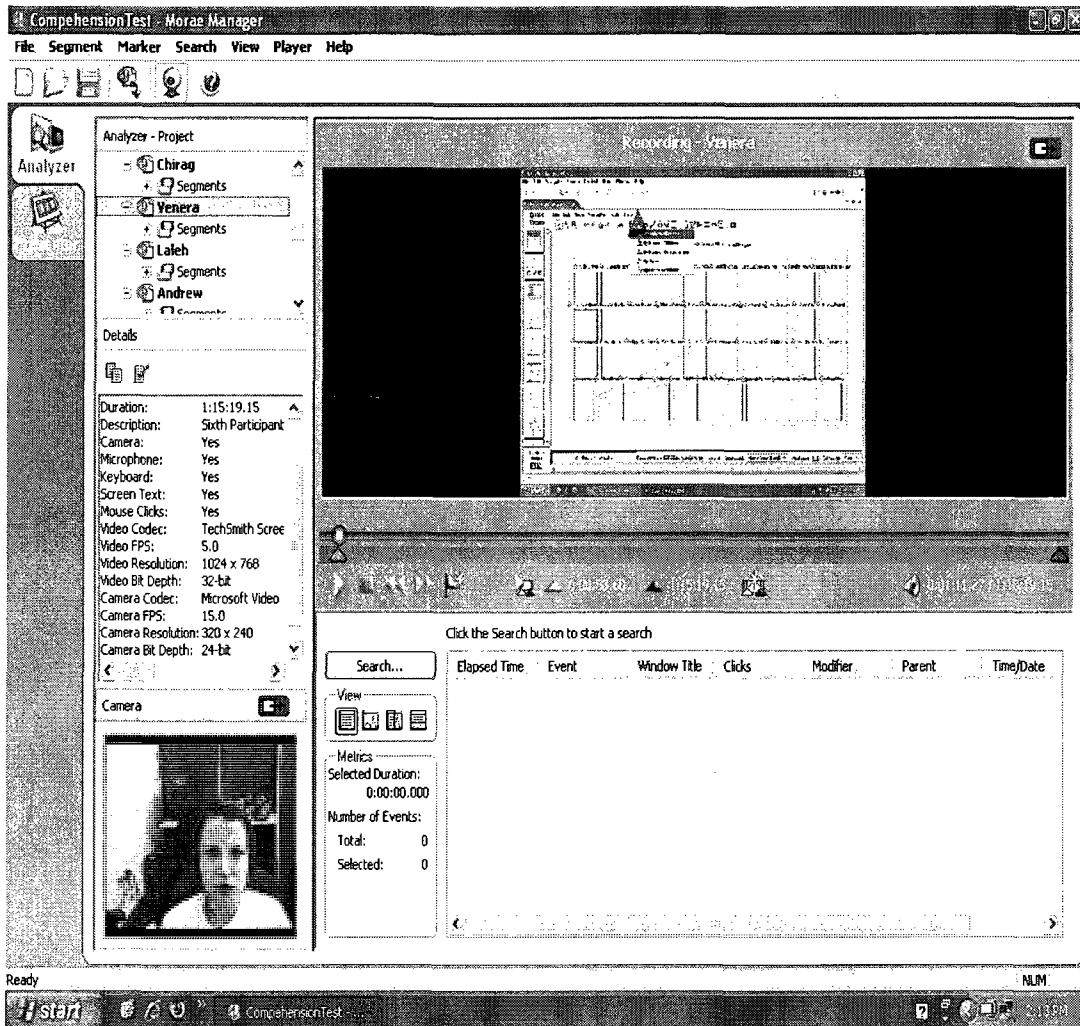


Figure 9.2: Analysis of Test Session with Morae Manager

Table 9.9 shows the comparative analysis of each technique based on the response time for a given task and number of mouse clicks to perform that task, which were generated using Morae manager. It also shows the order of techniques as tested by the participants and the comprehension value as assessed by our criteria along with the likeability of each participant.

Table 9.9: Comparative Analysis of Techniques

Participant #	Technique (order of evaluation)	Total Comprehension (in %)	Correctness of the task (Y: Yes) (N: No)	Response time (h:mm:ss.ms)	No. of mouse clicks	Likeability
1	Radial	87.50	Y	0:00:10.90	1	1. Tree
	NestedView	58.34	Y	0:01:36.75	13	2. Radial
	Pyramid	56.67	N	0:02:29.98	15	3. NestedView
	Tree	85.84	Y	0:00:16.97	3	4. Pyramid
2	NestedView	69.17	Y	0:01:59.48	55	1. Radial
	Radial	83.33	Y	0:00:35.35	6	2. Tree
	Tree	74.16	Y	0:01:08.60	45	3. NestedView
	Pyramid	64.16	N	0:01:31.17	27	4. Pyramid
3	Tree	67.50	Y	0:02:17.54	48	1. Radial
	NestedView	70.83	Y	0:00:39.44	9	2. Tree
	Pyramid	50.00	Y	0:03:22.69	48	3. NestedView
	Radial	80.00	Y	0:00:29.95	7	4. Pyramid
4	Pyramid	64.16	Y	0:01:30.48	28	1. Radial
	Tree	77.50	Y	0:01:11.95	20	2. Tree
	NestedView	81.67	Y	0:00:42.34	19	3. NestedView
	Radial	82.50	Y	0:00:13.70	3	4. Pyramid
5	Tree	56.67	N	0:01:19.40	18	1. Radial
	Radial	74.17	Y	0:00:13.45	1	2. NestedView
	NestedView	60.83	Y	0:01:35.28	24	3. Tree
	Pyramid	62.50	N	0:01:11.46	9	4. Pyramid
6	NestedView	69.17	Y	0:00:21.70	3	1. Radial
	Pyramid	60.00	Y	0:00:51.72	6	2. Tree
	Radial	85.00	Y	0:00:11.67	1	3. Pyramid
	Tree	72.5	Y	0:00:12.19	2	4. NestedView

Table 9.9 (continued)

Participant #	Technique (order of evaluation)	Total Comprehension (in %)	Correctness of the task (Y: Yes) (N: No)	Response time (h:mm:ss.ms)	Number of mouse clicks	Likeability
7	Pyramid	49.17	Y	0:04:27.56	35	1. Radial
	Radial	83.33	Y	0:00:16.59	3	2. Tree
	NestedView	65.84	N	0:00:22.94	5	3. NestedView
	Tree	70.01	Y	0:00:20.97	5	4. Pyramid
8	NestedView	80.83	Y	0:01:58.07	18	1. Radial
	Tree	80.83	Y	0:02:10.57	64	2. Tree
	Pyramid	64.16	Y	0:02:13.69	40	3. NestedView
	Radial	84.16	Y	0:00:04.40	1	4. Pyramid
9	Tree	58.33	Y	0:01:55.77	49	1. Radial
	NestedView	75.00	Y	0:00:15.18	2	2. Tree
	Radial	95.00	Y	0:00:03.07	1	3. NestedView
	Pyramid	43.33	Y	0:02:27.60	39	4. Pyramid
10	NestedView	65.84	Y	0:00:47.80	5	1. NestedView
	Radial	68.33	Y	0:00:12.40	1	2. Tree
	Pyramid	50.00	Y	0:02:49.68	3	3. Radial
	Tree	69.17	Y	0:00:04.68	1	4. Pyramid
11	Tree	78.34	Y	0:04:27.10	88	1. NestedView
	NestedView	81.67	Y	0:00:38.04	10	2. Tree
	Pyramid	81.67	Y	0:01:12.82	13	3. Radial
	Radial	85.00	Y	0:00:12.18	2	4. Pyramid
12	NestedView	63.33	Y	0:02:19.07	32	1. Radial
	Pyramid	60.84	Y	0:04:05.69	26	2. Tree
	Radial	95.00	Y	0:00:08.55	1	3. NestedView
	Tree	85.83	Y	0:00:28.53	4	4. Pyramid
13	Radial	56.67	Y	0:00:08.52	1	1. Tree
	Pyramid	36.67	Y	0:00:20.76	5	2. Radial
	Tree	65.01	Y	0:00:03.67	2	3. NestedView
	NestedView	53.34	Y	0:00:17.04	7	4. Pyramid

Table 9.9 (continued)

Participant #	Technique (order of evaluation)	Total Comprehension (in %)	Correctness of the task (Y: Yes) (N: No)	Response time (h:mm:ss.ms)	Number of mouse clicks	Likeability
14	Radial	69.16	Y	0:01:17.05	18	1. Tree
	Pyramid	45.83	Y	0:01:44.95	17	2. Radial
	Tree	60.84	N	0:00:54.20	30	3. NestedView
	NestedView	79.16	Y	0:00:10.97	1	4. Pyramid
15	Radial	71.67	Y	0:01:00.36	5	1. Radial
	Tree	71.67	Y	0:03:18.07	68	2. Pyramid
	NestedView	72.49	Y	0:00:46.93	7	3. NestedView
	Pyramid	55.83	N	0:05:23.71	50	4. Tree

In Table 9.9, we can see that some of the participants had trouble in correctly performing the assigned task (i.e. to find a class ‘MultiView’ and related information objects). Out of 15 participants, all (100%) were able to perform the asked task using the Radial technique, 11 (73.33%) participants performed the task correctly with Pyramid technique, 14 (93.33%) participants completed the task with NestedView technique, and 13 (86.67%) participants carried out the task completely with Tree technique.

Brief descriptions for the incorrect task performances are as under.

- Participants # 1, #2, #5, and #15 were not able to complete their assigned task with the ‘Pyramid’ technique correctly. They quitted before actually finding the asked class in the pyramid of other information objects.
- The participants # 5 and #14 changed the layout of visual representation from ‘Tree’ technique to ‘NestedView’ technique for this task. Therefore, the task time for Tree technique for participants # 5 and #14 is also not correct.

- The participant # 7 mistakenly selected the wrong class while finding the required class using the 'NestedView' technique.

Thus, we believe that the task time spent for the incomplete task as listed in Table 9.9 for these participants is not signifying the correct relationship with the comprehensibility of corresponding visualization technique as assessed through the questionnaire. Under this belief, we assume that actual task time for performing the complete task is more than this time, which then correctly portrays a relationship with the comprehensibility of corresponding visualization technique.

Verifying the hypothesis H1

In order to validate the responses obtained through our questionnaire with the scores obtained using objective metrics, we verified our defined hypothesis H1 (given in section 9.2.3) as shown in Table 9.10. In this table, we can see a clear relationship between the participants' comprehensibility of underlying visualization techniques as assessed by our questionnaire and their task performance with these visual representations except for three participants (i.e. participant # 2, #5, and #14). However, these three participants are those who were not able to complete their assigned task for some of the techniques as explained earlier. Therefore, we believe that their task times are inaccurate to consider for verifying our premise. Thus, we believe that our criteria seem to be able to give fairly accurate indication on the comprehensibility of each visual system for each of the participants with respect to this specific task.

The relationship between the comprehensibility and task time is not linear, as still there are a number of other external variables (outside the scope of this research) that are influencing this relationship. For example, comprehension is an individual's total

property of a visualization system and this can be accurately estimated by considering all the tasks that are supported by the visualization system. Moreover, there is one aspect ('Information Structure') of comprehension that is also beyond the scope of our evaluation and it may have a significant influence on the task performance.

Table 9.10: Verification of Hypothesis

Participant#	R → Radial Technique, P→ Pyramid Technique, N→ NestedView Technique, and T→ Tree Technique		
	Observed order for Comprehension score	Observed order of task time for each technique	Accept (✓) or Reject (×) Hypothesis (H1)
1	R>T>N>P	R<T<N<P	✓
2	R>T>N>P	R<T<P<N	×
3	R>N>T>P	R<N<T<P	✓
4	R>N>T>P	R<N<T<P	✓
5	R>P>N>T	R<P<T<N	×
6	R>T>N>P	R<T<N<P	✓
7	R>T>N>P	R<T<N<P	✓
8	R>T=N>P	R<N<T<P	✓
9	R>N>T>P	R<N<T<P	✓
10	T>R>N>P	T<R<N<P	✓
11	R>N=P>T	R<N<P<T	✓
12	R>T>N>P	R<T<N<P	✓
13	T>R>N>P	T<R<N<P	✓
14	N>R>T>P	N<T<R<P	×
15	N>R=T>P	N<R<T<P	✓

The comprehensibility of each visualization technique along with the time taken for the formal task is plotted in Figure 9.3 for each participant. From this figure, one can see

that for each participant the task time (except for incorrect tasks) is inversely proportional to the comprehensibility as assessed through our criteria, i.e. more the task time less is the comprehension support of corresponding visualization technique to the participant. Based on these results, we can conclude that the proposed framework can help in correctly estimating the comprehensibility of an individual for a particular visualization technique.

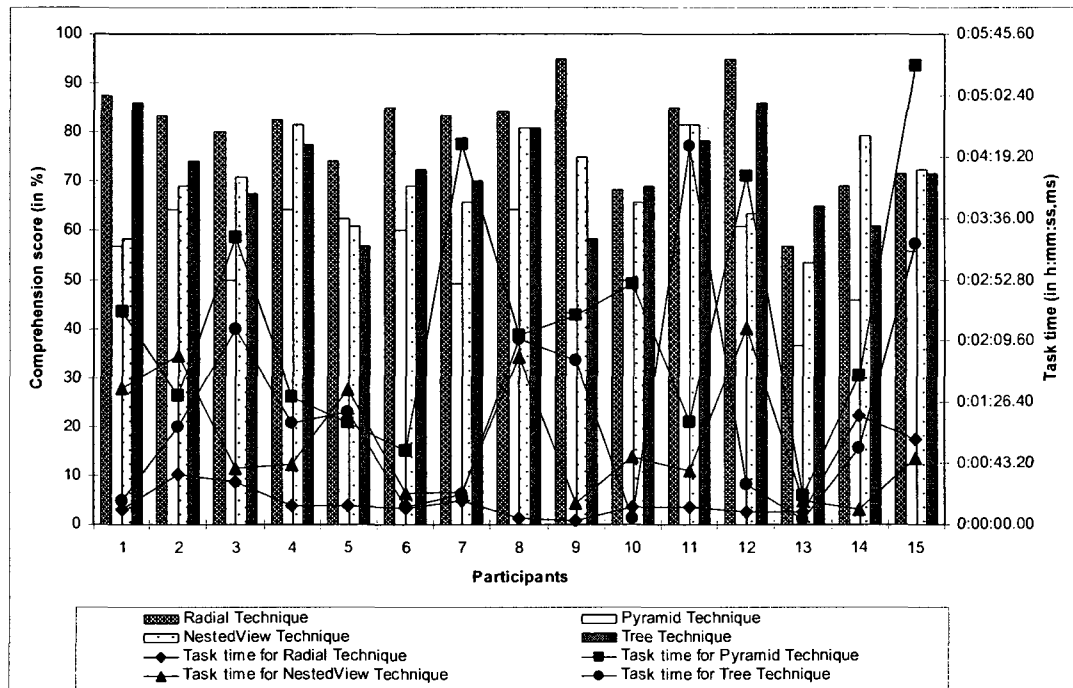


Figure 9.3: Plot of Comprehension Score and Task Time

9.3.5 Applicability of Criteria to Visualization Tools

Based on the comprehension problems reported by the participants during their tests with visualization techniques, we are summarizing the collective responses for each of the criteria as follows –

- **Applicability of Criteria to Structural Analysis for Java (SA4J)**

Reachability – The reachability in Radial technique was excellent, where the forward and backward buttons were clearly pointing where to look for specific objects in the

visualization. However, the reachability in Pyramid visualization was very poor, as the participants had no clue of the visited squares on the visualization. The participants did not know how to look for a particular information object on the Pyramid visualization. The 'Find' function for the pyramid visualization was not comprehensible by the participants.

Simplicity – Both the Radial and Pyramid visualizations were simple displaying only the necessary and relevant information on the screen. Some participants reported few redundancies in the menu bar options like – 'Project Wizard' option under the 'Option' menu item and 'New Java Project' option under the 'File' menu item were same. The participants also commented on the unutilized screen space in Pyramid visualization.

Clarity – Many of the icons on the interface for Radial technique were not clear to the participants. The most problematic icon was the 'Max Neighborhood' icon, which has a greyish background. The participants were falsely assuming that this icon is disabled. Other icons like – 'Hide Controls' was also not clear to the participants as they were thinking of it as opening another window object. For the Pyramid technique, the most problematic icon was the 'Skeleton' icon, which was like a run button. The participants were assuming that clicking it they were going to run some movie objects, but actually it was displaying to them the recently clicked information object.

Distinctiveness – For Radial technique, the visual attributes were helping the participants to identify each information object. However, for Pyramid technique this was not the case, as the same visual attributes were used to show classes and interfaces in the visualization.

Emphasis – This was good in both the Radial and Pyramid visualization techniques according to the feedback of participants.

Affordance – The visual clues to the functionality of some of the icons and symbols in both techniques were not clear at first glance. But, with the tool tip and after the initial exploration, the participants were able to comprehend the usage of these icons. The size of the squares in Pyramid technique was not clickable for all the participants.

Appearance – The layout of various information objects (i.e. packages, classes, and interfaces) in case of Radial visualization was revealing the features of underlying information. However, this was not easy in Pyramid technique as the relationships among the information objects were not directly visible.

Legibility – The screen was legible in terms of font size, font shape and color contrast.

Perspective-ness – The participants expressed the need to have other synchronized views to comprehend the whole software system. Most of the participants expressed the difficulty with Pyramid technique to find a specific information object.

Mapping – The interaction mechanisms along with the representations of information objects (i.e. classes, packages and interfaces) were quite natural to the participants.

Table 9.11 shows the average participants’ rating of comprehension criteria for SA4J.

Table 9.11: Rating of Criteria for SA4J

Visualization Technique	Criteria	Participants’ rating									
		Reachability	Simplicity	Clarity	Distinctiveness	Emphasis	Affordance	Appearance	Legibility	Perspective-ness	Mapping
Radial		93.33	88.88	76.67	73.33	100.0	73.33	56.67	90	60.00	88.33
Pyramid		55.55	66.67	63.33	48.33	80.00	70.00	28.33	91.11	10.00	50.00

- **Applicability of Criteria to Creole**

Reachability – The reachability in both the NestedView and Tree technique was average, where the forward and backward buttons were not working as intended. To look for a specific information object on visualization, the participants tried to use the provided search function. However, this function did not work out and the participants had to find the specific information object only by manually exploring the visualizations.

Simplicity – Both NestedView and Tree visualizations were simple displaying only the necessary and relevant information on the screen. The menu options were properly organized having related tasks together.

Clarity – The most problematic icon was the one used to arrange the visualization according to different styles like – alphabetical order, by number of children, by number of relationships and so on. This icon was not comprehensible to the participants.

Distinctiveness – For both the NestedView and Tree technique, the participants commented on the ambiguities of the icons in ‘Quick Views’ bar. Here, three icons representing ‘Package Dependencies via Field Accesses’, ‘Package Dependencies via Method Calls’, and ‘Package Dependencies via Method Calls Field Accesses’ were all shown with same visual attributes. Using the same icon for three different purposes was causing confusion to the participants.

Emphasis – This was good in both the NestedView and Tree visualization techniques according to the feedback of participants.

Affordance – The most problematic thing in Creole was that participants did not know how to change the layout in visualizations. They tried to change the layouts using the buttons provided on the interface. However, it was possible only through firstly selecting

the whole package and then clicking on the corresponding layout button. The manipulations in the visualizations were also not very easy as sometimes the vertical scrollbar was only partially visible.

Appearance – The layout of various information objects (i.e. packages, classes, and interfaces) in case of Tree visualization was revealing the features of underlying information. However, this was not easy in NestedView as the relationships among the information objects were sometimes crossing and getting cluttered.

Legibility – The screen was legible in terms of font size, font shape and color contrast for NestedView technique. However, this was somewhat problematic with Tree visualizations as many of information objects were overlapped and cluttered in one place.

Perspective-ness – The participants expressed the need to have other synchronized views to comprehend the whole software system.

Mapping – The interaction mechanisms along with the representations of information objects (i.e. classes, packages and interfaces) were quite natural to the participants.

Table 9.12 shows the average participants’ rating of comprehension criteria for Creole.

Table 9.12: Rating of Criteria for Creole

Visualization Technique	Criteria	Participants’ rating									
		Reachability	Simplicity	Clarity	Distinctiveness	Emphasis	Affordance	Appearance	Legibility	Perspective-ness	Mapping
NestedView		68.88	87.77	73.33	81.67	83.33	66.67	45.00	83.33	35.00	73.33
Tree		60.00	81.11	61.67	88.33	88.33	68.33	66.67	73.33	40.00	88.33

A comparative analysis of all four visualization techniques based on the participants' rating of comprehension criteria is shown in Figure 9.4.

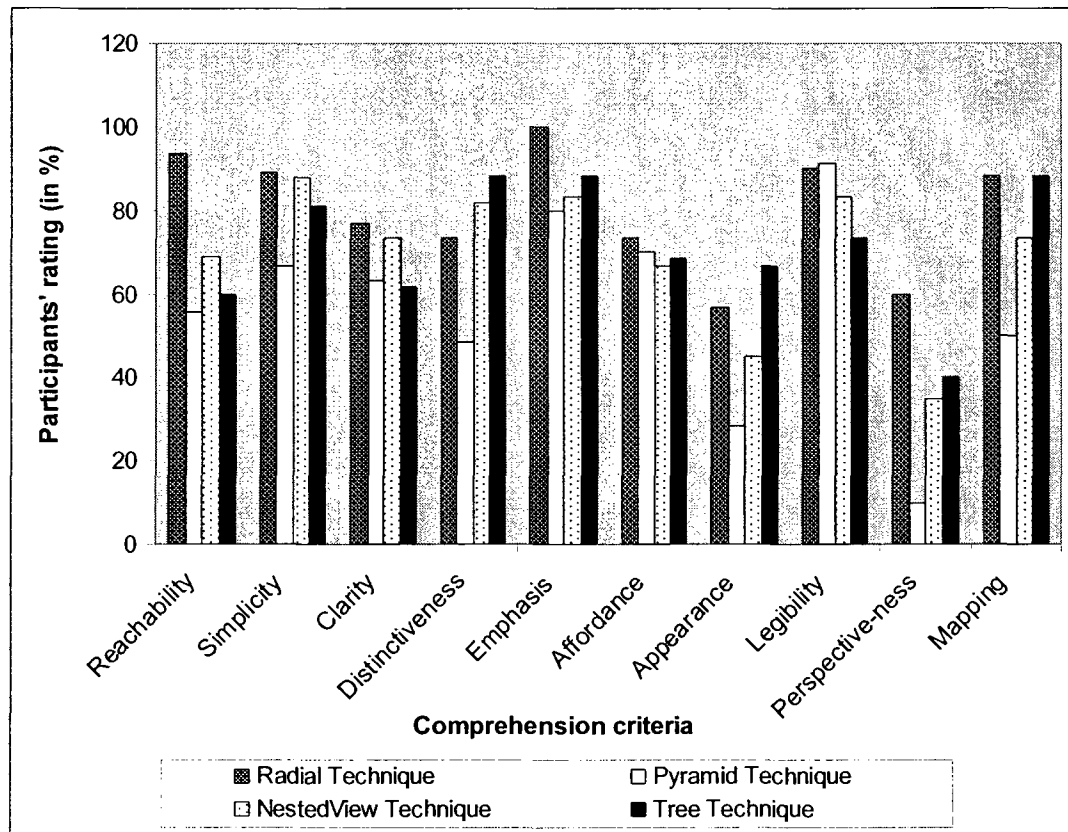


Figure 9.4: Rating of Criteria for Each Technique

A brief comparative analysis of each of the criteria based on the participants' rating is as under.

Reachability – The reachability in Radial technique scored highest (93.33%) among all the four visualization techniques. For the visualization techniques supported by Creole, NestedView was the one having higher rating than Tree visualization.

Simplicity – Both the Radial technique (88.88%) and NestedView technique (87.77%) were rated as simple, symmetric and well-organized.

Clarity – Radial (76.67%) and NestedView (73.33%) techniques were also rated as clear, where it was easy to identify various information objects despite any overlapping among them.

Distinctiveness – Tree (88.33%) and NestedView (81.67%) techniques were rated better than their counterparts for distinctiveness property i.e. these techniques have fewer ambiguities for the meanings of different icons/symbols displayed in them and information objects displayed in these visualizations are distinguishable.

Emphasis – Radial technique was rated as a technique where 100% emphasis is placed in the centre of the screen and the participants were intuitively focusing their attention on the main information object (i.e. a package).

Affordance – All the four visualization techniques were easing some cognitive load by using a certain set of affordances, where a highest value for affordance was observed for Radial Technique (73.33%).

Appearance – To depict the static structure of the software system, Tree technique (66.67%) was rated as the highest for the visual design, whereas the layout of Pyramid technique (28.33%) was rated as poorly designed among all four visualization techniques.

Legibility – The legibility in terms of font size, font type and color contrast was better in both the techniques (i.e. Radial and Pyramid) of SA4J (Structural Analysis for Java) tool than Creole's visualizations.

Perspective-ness – The need for having the perspective views was expressed for all the four visualization techniques, where again Radial technique (60%) was rated as an effective technique to fulfill the underlying task. The Pyramid technique (10%) was the most ineffective among all the four visualization techniques to fulfill the assigned task.

Mapping – The interaction styles with Radial and Tree technique were equally natural to the participants. The domain terminology was also familiar to the participants in these techniques.

9.3.6 Verifying the Null Hypothesis

The ANOVA analysis of the total comprehension score per participant of each visualization technique was computed to reject our null hypothesis (given in section 9.2.3). Through this analysis, we obtained an F ratio of 23.91 (computed in section H.2 of Appendix ‘H’), which is far greater than the F-critical value (i.e. F (3, 42)) of 4.29 at $p < 0.01$. Thus, it clearly demonstrates that all the four visualization techniques have different comprehensibility of the underlying information, with Radial technique having highest average comprehension score as shown in Figure 9.5. The score is averaged as all our participants are observed to be a homogenous group (based on ANOVA results in Section 9.3.2), and therefore their opinions are equally weighted.

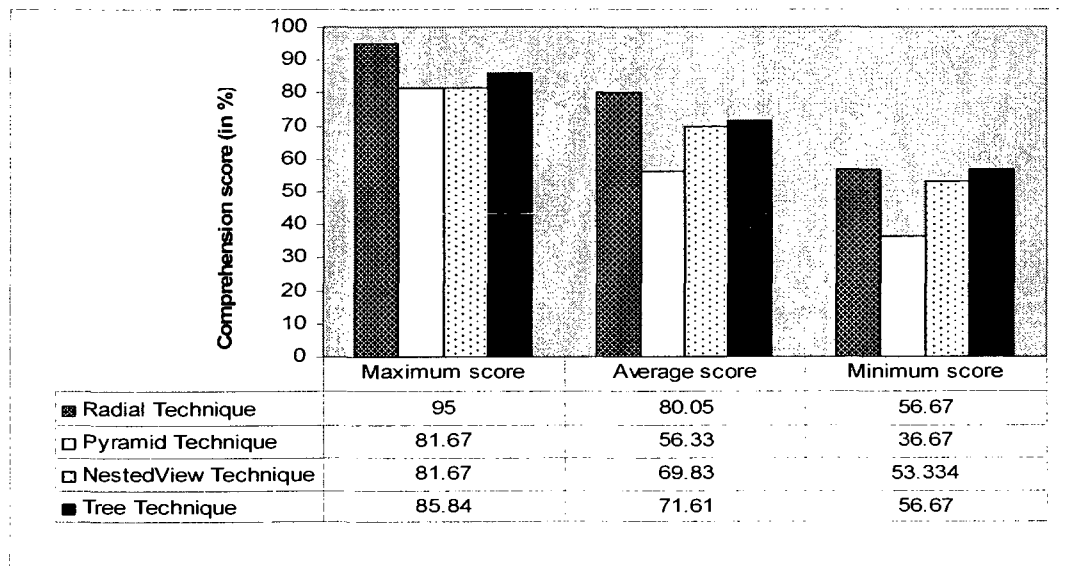


Figure 9.5: Comprehension Score of Each Technique

9.4 Discussion and Perspectives

Through this experiment, we have demonstrated the potential use of our framework to measure the comprehension support of visualization systems. We have seen that our framework is able to capture all the phases of a controlled experiment and is effectively gauging the comprehension support provided by the studied visualization systems. We have also observed that our proposed visualization patterns are a useful assistance mechanism to guide participants, as all our participants were able to understand the use of each visualization tool and their respective visualization techniques through the use of these patterns reasonably well.

Moreover, in this experiment we have witnessed a link between the responses of participants for the comprehension criteria and their actual task performances with the visualization techniques. We have also observed that overall satisfaction of the participants in terms of ‘likeability’ was highest for the Radial technique among all the four studied visualization techniques.

Using the statistical analysis techniques, we have seen that all the visualization techniques enable all the participants, irrespective of their expertise, to solve the simple exploration task i.e. there is not much variation in the comprehensibility of a novice and an expert user for these techniques. Furthermore, for these techniques, we also did not see any gender difference in the comprehension scores of our participants.

In order to produce more comprehensive validation results, more participants should be invited in the study. Moreover, a longer experiment time along with selective tasks is needed to truly capture the comprehension difficulties of the participants. We believe that to support a useful analysis at least two experimenters should be involved in the running

of the experiment. A concern with the current experiment design is that participants can learn from performing tasks with preceding visualizations techniques, influencing their performance with subsequent visualization techniques. One possible solution would have been to slightly alter the names of the information objects in the source code to mimic different visualizations.

Despite the above listed drawbacks and corresponding need for improvements in conducting usability experiments, we believe that these experiments show without doubt the usefulness of CoMoVA framework in systematic assessment of comprehension support provided by visualization systems.

Chapter 10. Conclusions, Contributions and Future

Avenues

“Reasoning draws a conclusion, but does not make the conclusion certain, unless the mind discovers it by the path of experience.” – Roger Bacon (1214-1294)

Overview

This chapter concludes the thesis work with a few concluding remarks, summary of significant contributions, benefits of this research, and potential avenues for further investigation.

10.1 Concluding Remarks

The main topic of this research has been measurement of the comprehension support provided by visualization systems, an intangible and seemingly immeasurable characteristic. Our journey has been a long and arduous one, but also a successful one. We have investigated a number of different disciplines including visual representations, human cognition, human computer interaction, visualization systems, software engineering, usability studies and measurement processes. We have adapted relevant ideas, principles, concepts, processes, and methods and formulated a novel framework to systematically assess in a quantitative fashion the comprehension support provided by a visualization system to its intended users. Such comprehension assessment can help to determine the effectiveness of visualization systems in providing users insights and understandings of the complex underlying artifacts represented through visual(s). This thesis is the first one to address this very fundamental characteristic of any visualization system. We have devised a hierarchical model in the form of:

- three factors/aspects (i.e. Presentation, Perception, and Cognition) that are involved in fully comprehending the presented visual information,
- eleven criteria (i.e. Reachability, Simplicity, Clarity, Distinctiveness, Emphasis, Affordance, Dynamism, Appearance, Legibility, Perspective-ness, and Mapping) that are the main building blocks of improving the visual comprehension, and
- related measures assessed empirically through suitably designed usability experiments.

The proposed set of criteria categorized according to different aspects in the communication of visual message is founded on the current work done in the field of perception, cognition and user interfaces by eminent researchers in HCI community.

Further, in this thesis, we propose a systematic evaluation mechanism in the form of visualization patterns that guides the tool users/evaluators to compare, understand and select appropriate visualization tools/techniques. Our approach for evaluation of visualization systems is similar to other questionnaire-based approaches such as – SUMI (Software Usability Measurement Inventory), where all the questions measure some properties of the common objective.

Empirical evaluation with appropriately crafted usability experiments on software visualization systems has demonstrated the veracity of our research hypothesis stated in the beginning of this thesis.

10.2 Contributions

The main contributions of this research are as follows –

1. A principal contribution of this research is the formulation of an empirical evaluation framework for systematically assessing comprehension support provided by a visualization system to its intended users. The proposed CoMoVA framework defines a clear protocol for controlled experimentation of visualization systems and provides a supporting structure that links various artifacts to deal with the measurement of comprehension in visualization systems.

In current practice, the evaluation of visualization systems is conducted in an ad-hoc manner without considering those fundamental characteristics of these systems that improve the understandings of their users. For example, the latest assessment of

visualization systems with heuristics (Zuk et al., 2006) covers three different perspectives and has no common focus for evaluation. This set of heuristics is not clearly defined and some of heuristics require domain expertise to understand and apply. However, in our framework, we are proposing a clear set of measurable attributes that are all focused on investigation of comprehension. It also requires no special expertise to apply the framework to assess comprehension support provided by any visualization system.

2. Earlier, the users/evaluators had no guidance mechanism to know the use of any visualization tool/technique and under which situations it was really useful. Through our proposed visualization patterns, the evaluators and users get a clear description of the problem that the visualization technique is addressing, the context in which it can be used and the design solution that it is supporting. With these patterns they can easily compare and understand the use of a visualization technique in a certain context.
3. In this research, we have also investigated in detail the needs of software maintainers and categorized them according to the four traditional maintenance activities. This thorough analysis can help to determine the success of current software visualization tools to fulfill the needs of software maintainers, i.e. the evaluators can seek the functional gap between the capabilities of existing tools/techniques and what is actually needed by the software maintainers. For example, in our studied software visualization systems, we found that only 50 to 60% of the maintenance tasks are supported by SA4J (Structural Analysis for Java) and Creole.

10.3 Research Benefits

We believe that our proposed research can contribute in many ways. We are highlighting five main benefits of measurement in general as under.

1. **Characterization/comparison:** Measurements produce objective results and convey the accurate view by empirically biasing some tool/technique with respect to other for some task. They can objectively tell us which tool/technique is more appropriate for a particular problem.
2. **Appropriateness:** The visualization tools/techniques are evaluated to judge their strengths and weaknesses. Measurements can show that a new visualization tool/technique is useful in a practical sense, according to the level of comprehension that can be achieved with it, for a specific task.
3. **Prediction:** A more fundamental goal of conducting measurements is to seek insight into why a particular visualization tool/technique is more effective. This can guide future efforts to improve existing tools/techniques. We want to understand the limitations of existing tools/techniques in terms of their supported tasks to comprehend the visuals presented through them. This knowledge is critical because we can guide developers to show multiple views or use multiple techniques where a single technique is not effective.
4. **Improvement:** Measures also help us plan and track improvement efforts. We need to be sure that new techniques are really better than old ones. Measures of current performance give us baselines to compare against, so that judgment can be made based on whether or not the improvement actions are working as intended and what the side effects are. Measurements show us how an abstract visualization design

theory applies under certain practical conditions. Measurement results can prove when the theories hold and how they need to be improved to function correctly for real-world data and tasks.

5. Supplementing experts' performance: A final use of measurements is to supplement expert users choice of a visualization tool/technique based on their expertise, with the measurement results derived from measures. Measurement will add the objective results with the subjective evaluation of expert users' performance.

In addition to these general benefits, our proposed framework will be a reusable solution that can be applied in real-world settings to measure comprehension. Specifically, we expect the following benefits from the use of our measurement framework:

1. Prior attention to the most important visual design principles for understanding what characteristics of a visualization system can influence users' comprehension.
2. Provide a flexible hierarchy of the factors, criteria and measures, so that evaluators could select those that are most appropriate according to their evaluation objectives.
3. Appropriate documentation of the test environment, in terms of 'context of use' and encapsulation template for visualization techniques in terms of visualization patterns, for better understanding and analysis.
4. During usability experiments, data collection efforts will be concentrated, since the required data elements are already defined.
5. Interpretation of data from usability experiments will be more efficient and effectively tied to selected objectives.

Therefore, we believe that our framework will be of maximum use to the software

community.

10.4 Future work

We believe that further applications of CoMoVA framework with additional systems would provide more evidence regarding its support for measuring comprehension. There are still a number of other avenues that can be explored further as follows –

- a) Validate the questionnaire – A questionnaire’ validity is the extent to which it measures what it claims to measure. A technique called ‘factor analysis’ that is normally applied in psychometric questionnaires evaluation can also be applied for the confirmatory analysis of our questionnaire. Factor analysis is a statistical procedure that examines the correlations among variables to discover clusters of related variables (Nunally, 1978). With the responses obtained from the participants, we can apply the multiple group method of the factor analysis technique to study the relationship among various criteria.
- b) Measurement scale to validate the values of measures – We have chosen a three point (i.e. ‘Yes’, ‘Somewhat’ and ‘No’) likert-scale for answering the questionnaire. However, we realized that having a seven point scale would produce more reliable results. Furthermore, more studies with other visualization systems are needed in order to define the threshold limits for the values of these measures.
- c) Inclusion of ‘Information Structure’ aspect – In this research, we did not consider ‘Information Structure’ aspect of comprehension. This aspect also impacts the accuracy of displayed visualizations, and therefore it needs further elaboration to determine the flaws in the data that can cause comprehension difficulties.

d) The software maintenance tasks identified in this thesis through literature review can be reused to create a standardized library of needs of the maintainers, which can help to identify the differences in the task support of any software visualization tool. This initial task model can also be refined further to include other elements like – the interaction, application and user tasks to capture the context of use in which the visualization tool can be used to support the required tasks.

References

- [1] Alam, S., Dugerdil, P., “EvoSpaces Visualization Tool: Exploring Software Architecture in 3D”, in *Proceedings of 14th Working Conference on Reverse Engineering (WCRE)*, Vancouver, Canada, 2007, pp: 269-270.
- [2] Alexander, C., Ishikawa, S., Silverstin, M., Jacobson, M., Fiksdahl-King, I., and Angel, S., *A Pattern Language – Towns, Buildings, Construction*, Oxford University Press, New York, 1977.
- [3] Anders, E.K., Kintsch, W., “Long-Term Working Memory”, *Psychological Review*, Volume 102, Issue 2, 1995, pp: 211-245.
- [4] Anslow, C., Marshall, S., Noble, J., and Biddle, R., “Software Visualization Tools for Component Reuse”, in *Proceedings of ACM Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA)*, Vancouver, Canada, October 2004.
- [5] Artho, C., Havelund, K., “Applying Jlint to Space Exploration Software”, *Lecture Notes in Computer Science (LNCS)*, Springer Berlin/Heidelberg, Volume 2937, 2003, pp: 61-75.
- [6] Ayama, M., Ujike, H., Iwai, W., Funakawa, M., and Okajima, K., “Effects of Contrast and Character Size upon Legibility of Japanese Text Stimuli Presented on Visual Display Terminal”, *Optical Review*, Volume 14, Issue 1, 2007, pp: 48–56.
- [7] Baecker, R.M., Grudin, J., Buxton, W., and Greenberg, S., *Readings in Human-Computer Interaction: Toward the Year 2000*, Morgan Kaufmann Publishers, San Francisco, CA, 1995.

- [8] Baker, P., Domik, G., Grinstein, G., Hewett, T.T., McGrath, M., and Owen, “ACM SIGGRAPH Curriculum for Visualization”, Editor: G. Domik., 2005. Available from:
<<http://wwwcs.uni-paderborn.de/fachbereich/AG/agdomik/visualisierung/vis-report/download/curriculum.pdf>> [Accessed July 04, 2006].
- [9] Baldonado, M.Q.W., Woodruff, A., and Kuchinsky, A., “Guidelines for Using Multiple Views in Information Visualization”, in *Proceedings of Working Conference on Advanced Visual Interfaces (AVI)*, Palermo, Italy, 2000, pp: 110-119.
- [10] Bartram, L., Ware, C., “Filtering and Brushing with Motion”, *Information Visualization*, Volume 1, Issue 1, 2002, pp: 66–79.
- [11] Basili, V.R., Rombach, H., “Tailoring the Software Process to Project Goals and Environments”, in *Proceedings of the 9th International Conference on Software Engineering*, 1987, pp: 345-357.
- [12] Bassil, S., Keller, R.K., “Software Visualization Tools: Survey and Analysis”, in *Proceedings of the 9th International Workshop on Program Comprehension (IWPC)*, Toronto, Canada, 2001, pp: 7-17.
- [13] Berg, F. F., Ahlstrom, U., “Evaluating Controller Use of Advanced Weather Products By Evaluating User Interaction Patterns”, in *Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting*, 2005, pp: 30-34.
- [14] Berger, M., BORG Calendar 1.6.1, 2007, Available from:
<http://mbcsoft.com/index.php?option=com_content&task=view&id=23&Itemid=38> [Accessed September 06, 2006].

- [15] Bertini, E., Santucci, G., “Quality Metrics for 2D Scatterplot Graphics: Automatically Reducing Visual Clutter”, in *Proceedings of 4th International Symposium on Smart Graphics*, Springer-Verlag, Lecture Notes in Computer Science, Volume 3031, 2004, pp: 77-89.
- [16] Blackwell, A.F., Britton, C., Cox, A., Green, T.R.G., Gurr, C., Kadoda, G., and Kutar, M.S., et al., “Cognitive Dimensions of Notations: Design Tools for Cognitive Technology”, in *Proceedings of Cognitive Technology*, 2001, pp: 325-341.
- [17] Bodart, F., Vanderdonckt, J., “Visual Layout Techniques in Multimedia Applications”, in *Proceedings of Conference on Human Factors in Computing Systems*, Boston, USA, 1994, pp: 121-122.
- [18] Boehm, B.W., Brown, J.R., Kaspar, H., Lipow, M., Macleod, G.J., and Merritt, M.J., *Characteristics of Software Quality*, North-Holland Publishing Company, New York, 1978.
- [19] Borchers, J. O., “A Pattern Approach to Interaction Design”, *International Conference on Designing Interactive Systems*, New York, USA, 2000, pp: 369-378.
- [20] BORG ranking from Sourceforge.NET, 2007, Available from: http://sourceforge.net/search/?type_of_search=soft&words=BORG [Accessed October 07, 2007].
- [21] Bramer, D.J., Scheitlin, T., Deardorff, R., Elliott, D., Hay, K., Marlino, M.R., Middleton, D., Pandya, R., Ramamurthy, M.K., Weingroff, M., and Wilhelmson, R.B., “Using an Interactive Java-Based Environment to Facilitate Visualization

- Comprehension”, in *Proceedings of 18th International Conference on Interactive Information and Processing Systems (IIPS)*, American Meteorological Society, Orlando, FL, 2002.
- [22] Brath, R., “Metrics for Effective Information Visualization”, in *Proceedings of IEEE Symposium on Information Visualization*, Washington, DC, USA, 1997, pp: 108 – 111(126).
- [23] Bugajska, M., “Framework for Spatial Visual Design of Abstract Information”, in *Proceedings of the 9th International Conference on Information Visualisation*, London, UK, 2005, pp: 713-723.
- [24] Burd, E., Overy, D., and Wheetman, A., “Evaluating Using Animation to Improve Understanding of Sequence Diagrams”, in *Proceedings of the 10th International Workshop on Program Comprehension (IWPC'02)*, June 2002, pp: 107 – 113.
- [25] Callendar, C., Creole, The CHISEL Group, University of Victoria, BC, Canada, 2006, Available from:
<<http://www.thechiselgroup.org/creole>> [Accessed February 04, 2008].
- [26] Card, S.K., Mackinlay, J.D., and Shneiderman, B., *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufman, Los Altos, California, 1999.
- [27] Card, S.K., Moran, T.P., and Newell, A., *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates Inc., New Jersey, USA, 1983.
- [28] Casner, S.M., “Task-analytic Approach to the Automated Design of Graphic Presentations”, *ACM Transactions on Graphics*, Volume 10, Issue 2, 1991, pp: 111-151.

- [29] Cattaneo, G., Faruolo, P., Ferraro-Petrillo, U., and Italiano, G.F., "JIVE: Java Interactive software Visualization Environment", *IEEE Symposium on Visual Languages and Human-Centered Computing (VL/HCC)*, Rome, Italy, September 2004, pp: 41-43.
- [30] Chall, J., "Readability: An Appraisal of Research and Application", *Bureau of Educational Research Monographs*, Issue 34, The Bureau of Educational Research Ohio State University, 1958.
- [31] Chapin, N., Hale, J.E., Khan, K.M., Ramil, J.F., and Tan, W.G., "Types of Software Evolution and Software Maintenance", *Journal of Software Maintenance Evolution: Research and Practice*, Volume 13, Issue 1, 2001, pp: 3-30.
- [32] Charters, S.M., Thomas, N., and Munro, M., "The End of The Line for Software Visualization? ", in *Proceedings of 2nd IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, Amsterdam, Netherlands, 2003, pp: 110-112.
- [33] Chedgey, C., Structure101, 2007, Available from:
<<http://www.headwaysoftware.com/products/structure101/index.php>> [Accessed February 04, 2008].
- [34] Chen, C., "Top 10 Unsolved Information Visualization Problems", *IEEE Computer Graphics and Applications*, Volume 25, Issue 4, July 2005, pp: 12 - 16.
- [35] Cioch, F.A., "Measuring Software Misinterpretation", *Journal of Systems Software*, Elsevier Science Publishing, 1991, pp: 85-95.

- [36] Cox, A., Fisher, M., and Muzzerall, J., "User Perspectives on a Visual Aid to Program Comprehension", in *Proceedings of 3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, Budapest, Hungary, 2005, pp: 70-75.
- [37] Craft, B., Cairns, P., "Beyond Guidelines: What Can We Learn from the Visual Information Seeking Mantra?", in *9th Annual International Conference on Information Visualisation*, Greenwich, UK, 2005.
- [38] Creole User Manual, The CHISEL Group, University of Victoria, BC, Canada, 2006, Available from:
<http://www.thechiselgroup.org/shrimp_manual> [Accessed February 04, 2008].
- [39] Cross II, J.H., Hendrix, T.D., Mathias, K.S., and Barowski, L.A., "Software Visualization and Measurement in Software Engineering Education: An Experience Report", in *29th Annual FRONTIERS IN EDUCATION CONFERENCE on Designing the Future of Science and Engineering Education*, San Juan, Puerto Rico, Volume 2, 1999, Available from:
<<http://fie.engrng.pitt.edu/fie99/papers/1288.pdf>> [Accessed July 04, 2005].
- [40] Cutmore, T. R. H., Hine, T. J., Maberly, K. J., Langford, N. M., and Hawgood, G., "Cognitive and Gender Factors Influencing Navigation in a Virtual Environment," *International Journal of Human-Computer Studies*, Volume 53, Issue 2, 2000, pp: 223-249.
- [41] Davidoff, J. B., *Cognition Through Color*, MIT Press, Cambridge, Massachusetts, 1991.

- [42] Davies, C., OSMOSE, Immersence, 1996, Available from:
<<http://www.immersence.com/index.htm>> [Accessed September 06, 2005].
- [43] Davies, C., "Virtual Space", *Space: In Science, Art and Society*, Editors: Penz, F., Radick, G., and Howell, R., Cambridge University Press, Cambridge, England, 2004, pp: 69-104.
- [44] Deelen, P., TraceVis, Department of Mathematics and Computing Science, Technische Universiteit Eindhoven, 2006, Available from:
<<http://www.win.tue.nl/~wstahw/projects/finished/PieterDeelen/index.html>>
[Accessed February 06, 2008].
- [45] Dondis, D.A., *A Primer of Visual Literacy*, The MIT Press, Cambridge, 1973.
- [46] Dudycha, D. J., "Principles of Map Design", Lecture notes on a course titled - Introduction to Cartography and Remote Sensing, Department of Geography, Faculty of Environmental Studies, University of Waterloo, Canada, 2003, Available from:
<<http://www.fes.uwaterloo.ca/crs/geog165/mapdesign.htm>> [Accessed July 17, 2004].
- [47] Ekenstierna, M., "Evaluation of User Assistance in Graphical User Interface Software", Masters thesis in Lund Institute of Technology, Lund University, 2002, Available from:
<http://serg.telecom.lth.se/education/master_theses/docs/_Rep.Ekenstierna.pdf>
[Accessed September 16, 2004].

- [48] Elm, W. C., Woods, D.D., “Getting Lost: A Case Study in Interface Design”, in *Proceedings of the Human Factors Society*, Santa Monica, CA, 1985, pp: 927-931.
- [49] Entin, E., Klare, G., “Relationships of Measures of Interest, Prior Knowledge, and Readability to Comprehension of Expository Passages”, *Advances in Reading/Language Research*, Volume 3, 1985, pp: 9–38.
- [50] Fenton, N.E., Pfleeger, S.L., *Software Metrics – A Rigorous & Practical Approach*, 2nd edition, Revised Printing, PWS Publishing Company, Boston, MA, 1997.
- [51] Ferweda, J. A., “Fundamentals of spatial vision”, Applications of Visual Perception in Computer Graphics, Course #32, *Association for Computing Machinery's Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH)*, 1998, pp: 1-27.
- [52] Firesmith, D., “Using Quality Models to Engineer Quality Requirements”, *Journal of Object Technology*, Volume 2, Issue 5, 2003, pp: 67-75.
- [53] Flores, N., Soares, D., Ferreira, H., and Rodrigues, M., “HotSpotter: a JavaML-based Approach to Discover Framework’s Hotspots”, in *Proceedings of XML: aplicações e tecnologias associadas (XATA)*, Braga, Portugal, February 2005.
- [54] Foley, J., Ribarsky, B., “Next-generation Data Visualization Tools”, *Scientific Visualization, Advances and Challenges*, Editors: Rosenblum, L., Earnshaw, R.A., et al., Academic Press, 1994.

- [55] Freimut, B., Punter, T., Biffel, S., and Ciolkowski, M., "State-of-the-Art in Empirical Studies", Technical Report ViSEK/007/E, Virtuelles Software Engineering Kompetenzzentrum (ViSEK), Version 1.0, 2001.
- [56] Friendly, M., "Gallery of Data Visualization", Statistical Consulting Service and Psychology Department, York University, 1999, Available from: <http://www.math.yorku.ca/SCS/Gallery/> [Accessed July 04, 2005].
- [57] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., *Design Patterns - Elements of Reusable Object Oriented Software*, Addison Wesley, Readings, MA, 1995.
- [58] Gärtner, J., Miksch, S., Carl-McGrath, S., "ViCo: A Metric for the Complexity of Information Visualizations", in *Proceedings of 2nd International Conference on Theory and Application of Diagrams*, Springer, Berlin, 2002, pp: 249-263.
- [59] Garvin, D.A., "What Does "Product Quality" Really Mean?" *Sloan Management Review*, Volume 26, Issue 1, 1984, pp: 25-43.
- [60] Georgiakakis, P., Psaromiligkos, Y., Retalis, S., "The Use of Design Patterns for Evaluating Personalisable Web-based Systems", in *Proceedings of 5th Workshop on User-Centered Design and Evaluation of Adaptive Systems*, Dublin, Ireland, June 2006, pp: 440-449.
- [61] Gershon, N., "Visualization of an Imperfect World", *IEEE Computer Graphics and Applications*, Volume 18, Issue 4, 1998, pp: 43-45.
- [62] Gershon, N., Eick, S.G., "Visualization Information", *IEEE Computer Graphics and Applications*, 1997, Available from: <http://www.cs.duke.edu/courses/spring03/cps296.8/papers/GuestEditor%27sInfoVisIntroduction.pdf> [Accessed September 16, 2004].

- [63] Ghoniem, M., Fekete, J.D., Castagliola, P., "A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations", *IEEE Symposium on Information Visualization*, Austin, Texas, USA, October 2004, pp: 17-24.
- [64] Gibson, J.J., *The Ecological Approach to Visual Perception*, Houghton Mifflin, Boston, 1979.
- [65] Gray, P. O., *Psychology*, 4th edition, Worth Publishers, 2001.
- [66] Grudin, J., "The Case Against User Interface Consistency", *Communications of ACM*, Volume 32, Issue 10, 1989, pp: 1164-1173.
- [67] Halpern, D. F., *Sex Differences in Cognitive Abilities*, 3rd edition, Lawrence Erlbaum Associates, Mahwah, NJ, 2000.
- [68] Haramundanis, K., "Learnability in Information Design", *ACM Special Interest Group for Design of Communication (SIGDOC'01)*, Santa Fe, New Mexico, USA, October 2001, pp: 7-11.
- [69] Healey, C. G., Booth, K. S., and Enns, J. T., "Visualizing Real-time Multivariate Data Using Pre-attentive Processing", *ACM Transactions on Modeling and Computer Simulation*, Volume 5, Issue 3, 1995, pp: 190-221.
- [70] Healey, C. G., Booth, K. S., and Enns, J. T., "High-speed Visual Estimation Using Preattentive Processing", *ACM Transactions on Computer-Human Interaction*, Volume 3, Issue 2, 1996, pp: 107-135.
- [71] Herisson, J., ADN-Viewer, LIMSI (Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur), 2001, Available from: <http://www.limsi.fr/> [Accessed July 10, 2004].

- [72] Hewett, T., "Extract from Cognitive Factors in Design: Basic Phenomena in Human Memory and Problem Solving", in *Proceedings of Computer Human Interaction on Cognitive Factors in Design*, 2003. Available from:
<<http://www.chi2003.org/docs/t08.pdf>> [Accessed September 16, 2004].
- [73] Hubona, G. S., Shirah, G. W., "The Gender Factor Performing Visualization Tasks on Computer Media," *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 4*, Big Island, HI, 2004.
- [74] IEEE Standard Computer Dictionary, A Compilation of IEEE Standard Computer Glossaries, New York, 1990.
- [75] IEEE Standard for Software Quality Metrics Methodology, IEEE Std 1061-1998, 1998, pp: 2-3, Available from:
<<http://csdl2.computer.org/comp//proceedings/hicss/2005/2268/01/22680029c.pdf>> [Accessed September 16, 2004].
- [76] IEEE, IEEE Standard for Software Maintenance (IEEE Std 1219-1998), Institute for Electrical and Electronics Engineers, NY, 1998.
- [77] Ingram, R., Benford, S., "The Application of Legibility Techniques to Enhance Information Visualizations", *The Computer Journal*, Volume 39, Issue 10, 1996, pp: 819-836.
- [78] Iskold, A., Kogan, D., Begic, G., Structural Analysis for Java (SA4J), An alphaworks Java technology from IBM, 2004, Available from:
<<http://www.alphaworks.ibm.com/tech/sa4j>> [Accessed 06 February 2008]
- [79] ISO 9126-1: Software Product Evaluation – Quality Characteristics and Guidelines for Their Use, ISO/IEC Standard, ISO-9126, International

- Organization for Standardization, Geneva, 1991 and Revised Draft, February 1997.
- [80] ISO 9126-1: Software Engineering - Product Quality – Part1: Quality Model, International Organization for Standardization, Geneva, 2001.
- [81] ISO/IEC 15939: Systems and Software Engineering - Measurement Process, International Organization for Standardization, Geneva, 2007.
- [82] IST, Inxight Star Tree: Illuminating Relationships, Networks and Large Information Hierarchies. Available from:
<<http://www.inxight.com/products/sdks/st/>> [Accessed July 28, 2005].
- [83] Jin, D., “Exchange of Software Representations among Reverse Engineering Tools”, External Technical Report, Department of Computing and Information Science, Queen's University, ISSN-0836-0227-2001-454, 2001, pp: 1-131.
- [84] Jones, C., *The Year 2000 Software Problem Quantifying the Costs and Assessing the Consequences*, Addison-Wesley, New York, USA, 1998.
- [85] Joshi, Y., “Assessment Criteria for Comprehensibility in Visualization Environments”, Masters Thesis, Department of Computer Science and Software Engineering, Concordia University, Canada, 2005.
- [86] Kane, L., Carthy, J., and Dunnion, J., “Readability Applied to Information Retrieval”, *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, Volume 3936, 2006, pp: 523-526.
- [87] Kapser, C.J., Godfrey, M.W., “Supporting the Analysis of Clones in Software Systems: A Case Study”, *Journal of Software Maintenance and Evolution: Research and Practice*, Volume 18, 2006, pp: 61–82.

- [88] Khelifi, A., Abran, A., and Buglione, L., “A System of References for Software Measurements with ISO 19761 (COSMIC-FFP)”, in *Proceedings of 14th International Workshop on Software Measurement (MetriKon)*, Berlin, Germany, 2004.
- [89] Kintsch, W., *Comprehension a paradigm for Cognition*, Cambridge University Press, New York, 1998.
- [90] Kintsch, W., Dijk, T.A.V., “Towards a Model of Text Comprehension and Production”, *Psychological Review*, Volume 85, 1978, pp: 363-394.
- [91] Kirakowski, J., Software Usability Measurement Inventory, 1996, Available from:
<<http://www.ucc.ie/hfrg/questionnaires/sumi/>> [Accessed September 19, 2005].
- [92] Kjelldahl, L., “A Survey of some Perceptual Features for Computer Graphics and Visualization”, in *Linköping Electronic Conference Proceedings of the Annual SIGRAD Conference*, Special Theme – Real-Time Simulations, Umeå University, Umeå, Sweden, November 2003.
- [93] Klemola, T., Rilling, J., “Modeling Comprehension Processes in Software Development”, in *Proceedings of the 1st IEEE International Conference on Cognitive Informatics*, 2002, pp: 329 – 336.
- [94] Klemola, T., Rilling, J., “A Cognitive Complexity Metric Based on Learning Category”, in *Proceedings of the 2nd IEEE conference on cognitive informatics*, 2003.
- [95] Knight, C., “Visualisation Effectiveness”, in *Proceedings of Workshop on Fundamental Issues in Visualisation*, Las Vegas, Nevada, USA, 2001.

- [96] Knight, C., Munro, M., “Visualisations; functionality and interaction”, in *International Conference on Computational Science*, Editors: Alexandrov, V. N., Dongarra, J., Juliano, B. A., Renner, R. S., and Tan, C. J. K., Springer, LNCS, San Francisco, CA, Volume 2074, 2001, pp: 470-475.
- [97] Kosara, R., Healey, C.G., Interrante, V., Laidlaw, D.H., and Ware, C., “User Studies: Why, How and When? ”, *Visualization Viewpoints: Column in IEEE Computer Graphics and Applications*, Editor: Rhyne, T. M., 2003, pp: 20-25, Available from:
<<http://www.cs.brown.edu/research/vis/docs/pdf/Kosara-2003-TUS.pdf>>
[Accessed July 20, 2005].
- [98] Koschke, R., “Software Visualization in Software Maintenance, Reverse Engineering, and Re-engineering: A Research Survey”, *Journal of Software Maintenance and Evolution: Research and Practice*, Volume 15, Issue 2, April 2003, pp: 87 – 109.
- [99] Koskinen, J., Salminen, A., and Paakki, J., “Hypertext Support for The Information Needs of Software Maintainers”, *Journal of Software Maintenance and Evolution: Research and Practice*, Volume 16, 2004, pp: 187–215.
- [100] Kosslyn, S.M., “Understanding Charts and Graphs”, *Applied Cognitive Psychology*, Volume 3, 1989, pp: 185-226.
- [101] Kosslyn, S.M., Gershon, N.D., Levkowitz, H., and Pearlman, J.D., “Improving Visualization: Theoretical and Empirical Foundations”, in *Proceedings of 3rd IEEE Visualization Conference*, Boston, Massachusetts, October 1992, pp: 372 – 374.

- [102] Kreitzberg, C., “User Centered Design Principles”, in *LUCID Framework* by Cognetics Corporation, 1998, Available from:
<<http://ei.cs.vt.edu/~cs3724/notes/lucid-appdx-a.pdf>> [Accessed November 17, 2004].
- [103] Kurniawan, S.H., “A Rule of Thumb of Icons' Visual Distinctiveness”, in *Proceedings of Conference on Universal Usability*, Arlington, Virginia, USA, 2000, pp: 159 – 160.
- [104] Lanza, M., “CodeCrawler - Polymetric Views in Action”, in *Proceedings of 19th IEEE international conference on Automated software engineering (ASE)*, 2004, pp: 394- 395.
- [105] Lewis, J. R., “IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instruction for Use”, *International Journal of Human-Computer Interaction*, Volume 7, Issue 1, 1995, pp: 57-78.
- [106] Lin, H.X., Choong, Y.Y., and Salvendy, G., “A Proposed Index of Usability: A Method for Comparing the Relative Usability of Different Software Systems”, *Behaviour and Information Technology*, Volume 16, Issue 4, 1997, pp: 267-278.
- [107] Lintern, R., Michaud, J., Storey, M. A., and Wu, X., “Plugging-in Visualization: Experiences Integrating a Visualization Tool with Eclipse”, in *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS)*, San Diego, USA, June 2003, pp: 47-56 (209).
- [108] Lowe, R.K., “Extracting Information from an Animation during Complex Visual Learning”, *European Journal of Psychology of Education*, Volume 14, Issue 2, 1999, pp: 225-244.

- [109] Lowe, R.K., “Animation and Learning: Selective Processing of Information in Dynamic Graphics”, *Learning and Instruction*, Volume 13, Issue 2, 2003, pp: 157-176 (20).
- [110] Lukoit, K., Wilde, N., Stowell, S., and Hennessey, T., “TraceGraph: Immediate Visual Location of Software Features”, in *Proceedings of International Conference on Software Maintenance (ICSM)*, 2000, pp: 33 - 39 .
- [111] Luzzardi, P.R.G., Frietas, C.M.D.S., Cava, R.A., Duarte, G.D., and Vasconcelos, M.H.S., “An Extended Set of Ergonomic Criteria for Information Visualization Techniques”, in *Proceedings of 7th IASTED Conference on Computer Graphics and Imaging (CGIM)*, Kauai, Hawaii, USA, 2004, pp: 144-152.
- [112] Ma, K.L., “Visualization: A Quickly Emerging Field”, *VISFILES: Column in ACM SIGGRAPH Computer Graphics*, Volume 38, Issue 1, 2004, pp: 4 – 7.
- [113] Mack, R. L., Nielsen, J., “Usability Inspection Methods: Executive Summary”, *Readings in Human-Computer Interaction: Toward the Year 2000*, 2nd edition, Editors: Baecker, R.M., et al., Morgan Kaufmann Publishers, 1995, pp: 170- 181.
- [114] Mackinlay, J., “Automating the Design of Graphical Presentations of Relational Information”, *ACM Transactions on Graphics*, Volume 5, Issue 2, April 1986, pp: 110-141.
- [115] Maletic, J. I., Marcus, A., Collard, M.L., “A Task Oriented View of Software Visualization”, in *Proceedings of 1st International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, Paris, France, 2002, pp: 32-40.

- [116] Marcus, A., “Managing Metaphors for Advanced User Interfaces”, in *Proceedings of the Workshop on Advanced Visual Interfaces*, Bari, Italy, 1994, pp: 12-18.
- [117] Marcus, A., “Principles of Effective Visual Communication for Graphical User Interface Design”, *Human Computer Interaction – Towards the year 2000*, 2nd edition, Morgan Kaufmann, San Francisco, 1995, pp: 425-441.
- [118] Marcus, A., “Metaphor Design in User Interfaces”, *Journal of Computer Documentation*, Volume 22, Issue 2, May 1998, pp: 43-57.
- [119] Marcus, A., Comorski, D., and Sergeyev, A., “Supporting the Evolution of a Software Visualization Tool Through Usability Studies”, in *Proceedings of International Workshop on Program Comprehension (IWPC)*, St. Louis, MO, 2005, pp: 307-316.
- [120] Mayrhauser, A.V., Vans, A.M., “Program Understanding Needs During Corrective Maintenance of Large Scale Software”, in *Proceedings Computer Software and Applications Conference COMPSAC*, IEEE Computer Society Press: Los Alamitos CA, 1997, pp: 630-637.
- [121] Mayrhauser, A.V., Vans, A.M., “Program Understanding During Software Adaptation Tasks”, in *Proceedings of International Conference on Software Maintenance (ICSM)*, IEEE Computer Society Press: Los Alamitos CA, 1998, pp: 316-325.
- [122] Mayrhauser, A.V., Vans, A.M., and Howe, A.D., “Program Understanding Behaviour during Enhancement of Large-scale Software”, *Journal on Software Maintenance: Research and Practice*, Volume 9, Issue 5, 1997, pp: 299–327.

- [123] McCall, J.A., Richards, P.K., and Walters, G.F., “Factors in Software Quality”, Rome Air Force Development Center, Technical Report *RADC TR-77-363*, Griffis Air Force, Rome, New York, 1977.
- [124] McGavin, M., Wright, T., and Marshall, S., “Visualisations of Execution Traces (VET): An Interactive Plugin-based Visualisation Tool”, in *Proceedings of Australasian User Interface Conference*, ACM International Conference Proceeding Series (50) 2006, pp: 153 – 160.
- [125] Merriam Webster Dictionary, 2007, Available from:
<<http://www.m-w.com/home.htm>> [Accessed August 03, 2004].
- [126] Miller, N., Hetzler, B., Nakamura, G., and Whitney, P., “The Need for Metrics in Visual Information Analysis”, in *Proceedings of the Workshop on New Paradigms in Information Visualization and Manipulation (NPIV)*, Las Vegas, USA, 1997, pp: 24 – 28.
- [127] Mohnkern, K., “Visual Interaction Design: Beyond the Interface Metaphor”, *SIGCHI Bulletin*, Volume 29, Issue 2, 1997, Available from:
<<http://sigchi.org/bulletin/1997.2/vid.html>> [Accessed November 22, 2004].
- [128] Mori, G., Paternò, F., and Santoro, C., “CTTE: Support for Developing and Analyzing Task Models for Interactive System Design”, in *Proceedings of Transactions in Software Engineering (TSE)*, Volume 28, Issue 9, 2002.
- [129] Mukherjea, S., Stasko, J.T., “Applying Algorithm Animation Techniques for Program Tracing, Debugging, and Understanding”, in *Proceedings of the 15th International Conference on Software Engineering*, May 1993, pp: 456 – 465.

- [130] Nakakoji, K., Takashima, A., and Yamamoto, Y., "Cognitive Effects of Animated Visualization in Exploratory Visual Data Analysis", in *Proceedings of 5th International Conference on Information Visualisation*, 2001, pp: 77-84.
- [131] Narayan, N.H., "Model-Based Hypermedia Design: Using Cognitive Models of Multimodal Information Comprehension to Design Hypermedia Visualizations", Position Paper for *CHI 97 Basic Research Symposium*, 1997, Available from:
<<http://www.eng.auburn.edu/users/narayan/brs97.html>> [Accessed September 16, 2004].
- [132] Nielsen, J., *Usability Engineering*, AP Professional Press, Boston, 1994.
- [133] Nielsen, J., "Top Ten Mistakes in Web Design", in Alertbox Column, 1997, Available from:
<<http://www.useit.com/alertbox/9706b.html>> [Accessed September 16, 2004].
- [134] Norman, D.A., *The Design of Everyday Things*, Doubleday, New York, 1990.
- [135] Norman, D.A., *Things That Make Us Smart: Defending Human Attributes in the Age of the Machine*, Perseus, Reading, Massachusetts, 1993.
- [136] North, C., "Toward Measuring Visualization Insight", *IEEE Computer Graphics and Applications*, Volume 26, Issue 3, 2006, pp: 6-9.
- [137] North, C., Shneiderman, B., "A Taxonomy of Multiple Window Coordinations", University of Maryland Computer Science Department, Technical Report, #CS-TR-3854, 1997.
- [138] Nunally, J.C., *Psychometric Theory*, McGraw-Hill, New York, 1978.
- [139] OED, Oxford English Dictionary, Available from:
<<http://www.oed.com/>> [Accessed July 04, 2005].

- [140] Orso, A., Jones, J., and Harrold, M.J., “Visualization of Program Execution Data for Deployed Software”, in *Proceedings of the ACM symposium on Software visualization*, San Diego, California, 2003, pp: 67 - 75.
- [141] Owen, G.S., “HyperVis - Teaching Scientific Visualization Using Hypermedia”, A project of the ACM SIGGRAPH Education Committee, the National Science Foundation (DUE-9752398), (DUE 9816443) and the Hypermedia and Visualization Laboratory, Georgia State University, 1999, Available from: <http://www.siggraph.org/education/materials/HyperVis/visgoals/visgoal2.htm> [Accessed September 16, 2004].
- [142] Pacione, M.J., “Effective Visualisation for Comprehending Object-Oriented Software: A Multifaceted, Three-Dimensional Abstraction Model for Software Visualisation”, Technical Report #EFoCS-52-2004, Department of Computer and Information Sciences, University of Strathclyde, Glasgow, UK, 2004, Available from: <http://www.cis.strath.ac.uk/~efocs/home/Research-Reports/EFoCS-52-2004Screen.pdf> [Accessed July 04, 2005].
- [143] Pacione, M.J., Roper, M., and Wood, M., “A comparative evaluation of dynamic visualisation tools”, in *Proceedings Working Conference on Reverse Engineering (WCRE)*, 2003, pp: 1095-1350.
- [144] Pacione, M.J., Roper, M., Wood, M., “A Novel Software Visualization Model to Support Software Comprehension”, in *Proceedings of 11th Working Conference on Reverse Engineering (WCRE)*, Delft, Netherlands, 2004, pp: 70-79.

- [145] Padda, H.K., "QUIM map: A Repository for Usability/Quality in Use Measurement", Masters thesis, Department of Computer Science, Concordia University, Montreal, 2003.
- [146] Pauw, W.D., Vlissides, J., "Visualizing Object-Oriented Programs with Jinsight", *Lecture Notes in Computer Science*, Springer, Berlin/Heidelberg, Volume 1543, 1998, Available from:
<<http://www.alphaworks.ibm.com/tech/jinsight>> [Accessed February 06, 2008].
- [147] Petre, M., Blackwell, A.F., and Green, T.R.G., "Cognitive Questions in Software Visualisation", *Software Visualization, programming as a Multi-Media Experience*, Editors: Stasko, J., Domingue, J., Brown, M., and Price, B., MIT Press, 1998, pp: 453-480.
- [148] Pfitzner, D., Hobbs, V., and Powers, D., "A Unified Taxonomic Framework for Information Visualization", in *Proceedings of the Australian symposium on Information visualization*, Adelaide, Australia, Volume 24, 2003, pp: 57 – 66.
Available from:
<<http://crpit.com/confpapers/CRPITV24Pfitzner.pdf>> [Accessed July 04, 2005].
- [149] Pfleeger, S.L., *Software Engineering Theory and Practice*, 2nd edition, Prentice Hall, 2001.
- [150] Pinker, S., "A Theory of Graph Comprehension", *Artificial intelligence and the future of testing*, Editor: R. Freedle, Lawrence Erlbaum Associates Inc., Hillsdale, N.J., 1990, pp: 73-126.

- [151] Plaisant, C., “The Challenge of Information Visualization Evaluation”, in *Proceedings of the Working Conference on Advanced Visual Interfaces*, Gallipoli, Italy, 2004, pp: 109 - 116.
- [152] Pressman, R.S., *Software Engineering - A Practitioner's Approach*, 6th edition, McGraw-Hill, New York, 2005.
- [153] Price, B.A., Baecker, R., and Small, I.S., “A Principled Taxonomy of Software Visualization”, *Journal of Visual Languages and Computing*, Volume 4, Issue3, 1998, pp: 211-266.
- [154] Purchase, H. C., Cohen, R. F., and James, M., “Validating Graph Drawing Aesthetics”, *Graph Drawing '95*, Lecture Notes in Computer Science, Springer-Verlag, volume 1027, 1996, pp: 435-446.
- [155] Raza, A., Vogel, G., Plodereder, E., “Bauhaus – A Tool Suite for Program Analysis and Reverse Engineering”, *Reliable Software Technologies, Ada Europe*, 2006.
- [156] Reed, S.K., *Cognition: Theory and Applications*, Brooks/Cole Publishing Company, 1996.
- [157] Reiss, S.P., “Dynamic Detection and Visualization of Software Phases”, in *Proceedings of Workshop on Dynamic Analysis (WODA)*, St Louis, MO, May 2005.
- [158] Rheingans, P., Landreth, C., “Perceptual Principles for Effective Visualizations”, *Perceptual issues in Visualization*, Editors: Grinstein, G., and Levkowitz, H., Springer-Verlag, 1995, pp: 59-74.

- [159] Robson, C., *Real Word Research: A Resource for Social Scientists and Practitioners- Researchers*, Blackwell, 1993.
- [160] Roberts, F.S., *Measurement Theory*, Addison-Wesley, Reading, MA, 1979.
- [161] Rombach, H.D., “Practical Benefits of Goal-Oriented Measurement”, *Software Reliability and Metrics*. Editors: Fenton, N., Littlewood, B., London, Elsevier Science Publishing Company, 1991, pp: 217-235.
- [162] Rushmeier, H., Botts, M., Uselton, S., Walton, J., Watkins, H., and Watson, D., “Panel: Metrics and Benchmarks for Visualization”, in *Proceedings of the 6th IEEE Visualization Conference (VISUALIZATION '95)*, pp: 422.
- [163] Saltz, J.S., Steinbach, J.M., “Understanding Information Visualizations”, *CODATA Euro-American Workshop on Data and Information Visualization: Where We Are and Where Do We Go From Here?* Paris, France, 1997, Available from:
<<http://www.codata.org/archives/1997Vis/sp4.htm>> [Accessed July 04, 2005].
- [164] Schiffman, H.R., *Sensation and Perception: An Integrated Approach*, 4th edition, John Wiley & Sons, 1996.
- [165] Schmidt, S. R., “Can we have a distinctive theory of memory? “, *Memory and Cognition*, Volume 19, 1991, pp: 523- 542.
- [166] Seaman, C., “Qualitative Methods in Empirical Studies of Software Engineering”, *IEEE Transactions on Software Engineering*, 1999, pp: 557-572.
- [167] Seffah, A., Donayee, M., Kline, R.B., Padda, H.K., “Usability Measurement: A Roadmap for a Consolidated Model”, *Software Quality Journal*, Springer Netherlands, Volume 14, Issue 2, June 2006, pp: 159 – 178.

- [168] Shneiderman, B., *Designing the user interface: Strategies for Effective Human-Computer Interaction*, 2nd edition, Addison-Wesley Publishing Company, MA, 1992.
- [169] Shneiderman, B., “The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations”, in *Proceedings of the IEEE Symposium on Visual Languages*, Boulder, USA, 1996, pp: 336-343.
- [170] Shneiderman, B., *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 3rd edition, Addison–Wesley, Reading, Massachusetts, 1998.
- [171] SmallWiki, A non-exhaustive list of Software Visualization tools. Software Composition Group, University of Bern, Switzerland, 2007, Available from: <http://smallwiki.unibe.ch/codecrawler/anon-exhaustivelistofsoftwarevisualizationtools/> [Accessed November 08, 2006].
- [172] Smolnik, S., Nastansky, L., Knieps, T., “Mental Representations and Visualization Processes in Organizational Memories”, in *Proceedings of the 7th International Conference on Information Visualization*, 2003.
- [173] Stavrinoudis, D., Xenos, M., Peppas, P., and Christodoulakis, D., “Early Estimation of Users’ Perception of Software Quality”, *Software Quality Journal*, Volume 13, 2005, pp: 155-175.
- [174] Storey, M.A.D., Čubranić, D., German, D.M., “On the Use of Visualization to Support Awareness of Human Activities in Software Development: A Survey and a Framework”, in *Proceedings of the ACM Symposium on Software Visualization*, St. Louis, Missouri, 2005, pp: 193-202 (216).

- [175] Storey, M.A.D., Müller, H.A., “Manipulating and documenting software structures using SHriMP views”, in *Proceedings of International Conference on Software Maintenance (ICSM)*, Opio (Nice), France, 1995, pp: 275- 284.
- [176] Storey, M.A.D., Wong, K., and Müller, H.A., “Rigi: A Visualization Environment for Reverse Engineering”, in *Proceedings International Conference on Software Engineering (ICSE)*, Boston, Massachusetts, USA, 1997, pp: 606-607, Available from:
<<http://www.rigi.csc.uvic.ca/index.html>> [Accessed December 10, 2007].
- [177] Storey, M.A.D., Wong, K., Fong, P., Hooper, D., Hopkins, K., and Müller, H.A., “On Designing an Experiment to Evaluate a Reverse Engineering Tool”, in *IEEE Proceedings of the 3rd Working Conference on Reverse Engineering (WCRE)*, Monterey, CA, 1996, pp.31-40.
- [178] Swanson, E.B., “The Dimensions of Maintenance”, in *Proceedings of International Conference on Software Engineering (ICSE)*, IEEE Computer Society, Long Beach, CA, 1976, pp: 492-497.
- [179] Systä, T., Koskimies, K., and Müller, H., “Shimba- an Environment for Reverse Engineering Java Software Systems”, *Software-Practice and Experience*, Wiley, New York, Volume 31, Issue 4, 2001, pp: 371-394.
- [180] Tan, J.K.H., Benbasat, I., “Processing of Graphical Information: A Decomposition Taxonomy to Match Data Extraction Tasks and Graphical Representations”, *Information Systems Research*, Volume 1, Issue 4, 1990, pp: 416-438.

- [181] Tidwell, J., *Designing Interfaces: Patterns for Effective Interaction Design*, O'Reilly Media, CA, USA, 2005, Available from:
<<http://time-tripper.com/uipatterns/>> [Accessed March 13, 2006].
- [182] Tonella, P., "Workshop on Empirical Studies in Reverse Engineering", in *Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice (STEP)*, 2005, pp: 61-64.
- [183] Torry, M., Möller, T., "Human Factors in Visualization Research", *IEEE Transactions on Visualization and Computer Graphics*, Volume 10, Issue 1, 2004, pp: 72-84.
- [184] Trafton, J.G., Kirschenbaum, S.S., Tsui, T.L., Miyamoto, R.T., Ballas, J.A., and Raymond, P.D., "Turning pictures into numbers: Extracting and Generating Information from Complex Visualizations", *International Journal of Human Computer Studies*, Volume 53, Issue 5, 2000, pp: 827-850.
- [185] Treemap, Human Computer Interaction Lab, University of Maryland, USA, 2003, Available from:
<<http://www.cs.umd.edu/hcil/treemap/>> [Accessed August 12, 2006].
- [186] Tufte, E.R., *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, CT, 1983.
- [187] Tufte, E.R., *Envisioning Information*, Graphics Press, Cheshire, CT, 1990.
- [188] Tufte, E.R., *Visual Explanations: Images and Quantities, Evidence and Narrative*, Graphics Press, Cheshire, CT, 1996.
- [189] Turner, J.R., Thayer, J. F., *Introduction to Analysis of Variance: Design, Analysis, & Interpretation*, Sage publications Inc., 2001.

- [190] Velez, M. C., Silver, D., Tremaine, M., “Understanding Visualization through Spatial Ability Differences”, in *Proceedings of the IEEE Visualization*, Minneapolis, MN, 2005, pp: 511-518.
- [191] Walenstein, A., “Cognitive Support in Software Engineering Tools: A Distributed Cognition Framework”, Doctoral dissertation, Computing Science Department, Simon Fraser University, Burnaby, B.C., Canada, 2002.
- [192] Walenstein, A., “Observing and Measuring Cognitive Support: Steps Toward Systematic Tool Evaluation and Engineering”, in *Proceedings of the 11th International Workshop on Program Comprehension*, 2003, pp: 185-195.
- [193] Ware, C., *Information Visualization: Perception for Design*, Morgan Kaufmann, San Francisco, 2000.
- [194] Welie, M., Veer, G.C., and Eliens, A., “Patterns as Tools for User Interface Design”, *International Workshop on Tools for Working with Guidelines*, Biarritz, France, 2000, pp: 313-324.
- [195] Wickens, C.D., *Engineering Psychology and Human Performance*, Harper Collins, New York, 1992.
- [196] Wijnholds, A.B., “Using Type: The Typographer’s Craftsmanship and the Ergonomist’s Research”, Utrecht University, April 1996, Available from: <<http://www.plainlanguagenetwork.org/type/typehimn.htm>> [Accessed August 20, 2006].
- [197] Wikipedia, Online encyclopedia, 2007, Available from: <http://en.wikipedia.org/wiki/Main_Page> [Accessed February 10, 2007].

- [198] Wilhelmson, R. B., Jewett, B. F., Shaw, C., Wicker, L. J., Arrott, M., Bushell, C. B., Bajuk, M., Thingvold, J., and Yost, J.B., "A Study of the Evolution of a Numerically Modeled Severe Storm", *International Journal of Supercomputing Applications*, Volume 4, Issue 2, 1990, pp: 20–36.
- [199] Wilkins, B., "MELD: A Pattern Supported Methodology for Visualisation Design", Doctoral dissertation, School of Computer Science, University of Birmingham, UK, 2003.
- [200] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslen, A., *Experimentation in Software Engineering - An Introduction*, The Kluwer International Series in Software Engineering, Kluwer Academic publishers, 2000.
- [201] Wolf, R., "Consistency as Process", *Coordinating User Interfaces for Consistency Checking*, Editor: Nielsen, J., London, Academic Press Inc., 1989, pp: 89-92.
- [202] Wünsche, B., "A Survey, Classification and Analysis of Perceptual Concepts and their Application for the Effective Visualisation of Complex Information", *Australasian Symposium on Information Visualisation*, Christchurch, New Zealand, Volume 35, 2004.
- [203] Wünsche, B., Lobb, R., "A Scientific Visualization Schema Incorporating Perceptual Concepts", in *Proceedings of Image and Vision Computing '01 New Zealand (IVCNZ '01)*, University of Otago, Dunedin, November 2001, pp: 31-36.
- [204] Zelkowitz, M. V., Wallace, D.R., "Experimental Models for Validating Technology", *IEEE Computer*, Volume 31, Issue 5, 1998, pp: 23-31.
- [205] Zhu, L., Babar, M.A., Jeffery, R., "Mining Patterns to Support Software Architecture Evaluation", in *Proceedings of Fourth Working IEEE/IFIP*

Conference on Software Architecture (WICSA), Oslo, Norway, June 2004, pp: 25-34.

[206] Zuk, T., Schlesier, L., Neumann, P., Hancock, M.S., and Carpendale, S., “Heuristics for Information Visualization Evaluation”, in *Proceedings of AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, Venice, Italy, 2006, pp: 1-6.

[207] Zuse, H., *A framework of Software Measurement*, Walter de Gruyter, Berlin, New York, 1998.

Appendix A. The Proposed Questionnaire

Overview

This questionnaire has been refined in three iterations with suggestions from our colleagues in human-centered software engineering lab. It was decided to use the same scheme for all the questions, and a detailed answer for the middle choice only to see the range of comprehension difficulties within the two extreme values ('Yes' for 100% to 'No' for 0% comprehensibility). This was also done in order to keep the question answering efforts minimum and within the time-constraints of a controlled experiment.

A.1 Glossary

Before presenting the questionnaire to the readers, we want to clarify the following terms used in this questionnaire.

Information objects:

Information elements displayed in the visualization, like – classes, packages, interfaces

Icons:

Icons are pictorial representations of screen objects, like a picture of a house icon meaning 'home'.

Symbols:

Symbols are signs, characters, or other concrete representations of ideas, concepts, or abstractions that represents something, such as '\$' is a symbol for dollars, '+' is a symbol for plus, and a flag is a symbol of a country.

A.2 The Questionnaire

The proposed questionnaire is designed to measure each comprehension criterion as follows.

Questions to measure *Reachability* criterion

1. Are you able to navigate from one location to another in the visualization?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

2. Are you able to undo your manipulation operations (e.g. select, click, move etc.) with the visualization to go back successfully to previously displayed screen?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

3. Are you able to see the location of any information object with respect to an overall context of other information objects in the display?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

Questions to measure *Simplicity* criterion

1. Does the organization of menus seem logical (i.e. are the related tasks grouped together)?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

2. Is the visualization symmetrically well-represented (i.e. organized vertically or horizontally) to utilize the screen space?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

3. Is there only necessary (i.e. non redundant, reasonable) information on the screen?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

Questions to measure *Clarity* criterion

1. Are you able to understand the meanings of icons/symbols/labels used in the display?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

2. Are you able to clearly identify the information objects displayed in the visualization despite any overlapping?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

Questions to measure *Distinctiveness* criterion

1. Are the used visual attributes (like - size, shape, color, texture etc.) for icons/information objects appropriate to distinguish them in display?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

2. Can the icons/symbols/labels be interpreted or expressed in only one way (i.e. there is no ambiguity in their meaning)?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

Questions to measure *Emphasis* criterion

1. Are you able to see the most important element in the display based on any visual attribute like - color, motion, shape, size, texture etc.?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

2. Are you able to differentiate (based on visible change in shape/color/size/brightness etc.) the object that you select from the one that is not selected?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

Questions to measure *Affordance* criterion

1. Is it easy to figure out how to use various artifacts (e.g. buttons, links, information objects, icons and so on) in the visualization based on the given visual cues?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

2. Are you able to easily manipulate (e.g., select, move, click etc.) all the artifacts (e.g. buttons, links, information objects, icons and so on) in the visualization?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

Questions to measure *Dynamism* criterion

1. Are you able to understand what is going on in the animation?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

2. Are you able to detect the location in the visualization where the critical changes occur?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

Questions to measure *Appearance* criterion

1. Just looking at the visualization, can you answer which information objects are related to one another?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

2. Do you think the visual design of the information objects reveals the features of the underlying information? (For example: size of a 'class' or depth of an information object in the information hierarchy)

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

Questions to measure *Legibility* criterion

1. Are the icons/labels readable?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

2. Is the color used for symbols/labels in good contrast to the background color?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

3. Is the font size and font type used for labels appropriate?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

Questions to measure *Perspective-ness* criterion

1. Do you think with this visualization you are able to effectively perform the intended tasks (Take an example: task 1 to show the related dependencies in a software system)?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

2. Do you think that there is no need of some other related views in the form of different visualizations to fully understand the underlying system?

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

Questions to measure *Mapping* criterion

1. Are the interaction methods of input devices, like – mouse, keyboard or vests in HMD etc., natural to you? (For example: if clicking the buttons on the mouse leads you to what you want to do with the visualization system?)

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

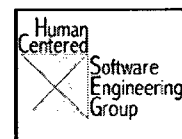
2. Are the icons/symbols used in the visualizations similar to the one that are used in the underlying domain? (For example: symbols used to depict relationships in UML.)

Yes	Somewhat	No
-----	----------	----

If somewhat, briefly explain why?

--

Appendix B. A Survey-based Empirical Investigation for Software Visualization



Informed Consent to Participate in Research

This is to state that I agree to participate in a research study on Developing a Framework to Measure Comprehension Support Provided by Visualization Systems conducted at the Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada. The main researchers are: Dr. Ahmed Seffah, Dr. Sudhir Mudur and Mrs. Harkirat Kaur Padda. Mrs. Padda, PhD candidate (padda@encs.concordia.ca, phone 514-848-2424 ext. 7165) is in charge of the study.

A. Purpose of the study

I have been informed that the purpose of the research is to rate the tasks supported by any software visualization tool to help in software maintenance activities. In addition, I will be asked to comment on other task(s) that in my opinion are suitable for software visualization tools and are not included in the list.

B. Procedure

The study is designed to ask software professionals having some experience with software visualizations or software maintenance to share their knowledge of software maintenance tasks supported/required of software visualization tools. You will be asked to answer a questionnaire comprising a list of tasks supported by current software visualization tools and rate them on a scale provided on questionnaire. In addition, you

are requested to comment on additional tasks that you may think are not added in the list and are also supported by software visualization tools for software maintenance purposes. This should take no more than 10 minutes of your time.

We anticipate no risk to you as a result of your participation in this study other than the inconvenience of the time to complete the questionnaire. As a result of your participation in this study, it is hoped that we may gain valuable information about the tasks supported by software visualization tools.

All provided information and data collection will be stored anonymously in a database, with no information that can identify participants. Results from the survey will be reported only in aggregate form in scientific communications like articles, workshops, and conference presentations.

C. Copyright

The questionnaire provided below is a copyright property of the researchers mentioned above and a reproduction of any sort is not allowed.

Name:

If you wish to receive the results of this study, please check the box below and provide an e-mail address

Mail address:

If at any time you have questions about your rights as a research participant, please contact Adela Reid, Research Ethics and Compliance Officer, Concordia University, at (514) 848-7481 or by email at areid@alcor.concordia.ca.

Questionnaire

Tasks	Required to accomplish which maintenance activity? (Tick all that apply)	Apply to which Software Visualization category? (Tick all that apply)	Rate the task in order of importance
1. Get the execution trace of source code	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
2. Get the static structure of the software system	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
3. Find the location of desired code segment	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
4. List of all artifacts that call a specific artifact	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
5. Determine the impact of changing an artifact without having to do it first	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
6. Does the run-time behavior contain regular repeated behavioral patterns?	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
7. When was an exception thrown or when did an error occur?	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
8. Find the location to insert a new artifact	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>

Questionnaire (continued)




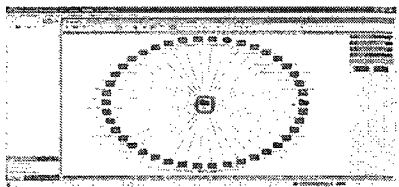
Tasks	Required to accomplish which maintenance activity? (Tick all that apply)	Apply to which Software Visualization category? (Tick all that apply)	Rate the task in order of importance
9. Add an artifact and dependencies (if any)	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
10. Find an artifact that is not used	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
11. Find an artifact that is heavily used	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
12. Determine which clusters of objects are closely related to one another, based on the amount of message traffic between them	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
13. Find identical coding pattern segments within the source code	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
14. What is the load on each component of the software system at runtime?	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
15. History of past modifications?	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
16. Nesting Level of a particular method	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>

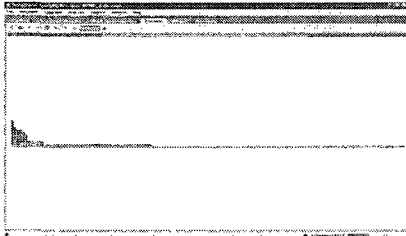
Questionnaire (continued)

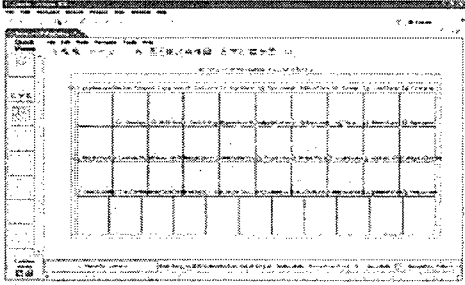
Tasks	Required to accomplish which maintenance activity? (Tick all that apply)	Apply to which Software Visualization category? (Tick all that apply)	Rate the task in order of importance
17. Where in the software system are the hotspots to add additional functionality?	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
18. Modify the artifact and dependencies(if any)	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
19.Delete an artifact and dependencies (if any)	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
20.Find an exact location to set a breakpoint	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>
21.Find all artifacts that directly or indirectly depend on artifact "A" or Find all artifacts on which artifact "A" directly or indirectly depends	Corrective <input type="checkbox"/> Adaptive <input type="checkbox"/> Perfective <input type="checkbox"/> Preventive <input type="checkbox"/>	Static <input type="checkbox"/> Dynamic <input type="checkbox"/>	Not important <input type="radio"/> Somewhat <input type="radio"/> Extremely important <input type="radio"/>

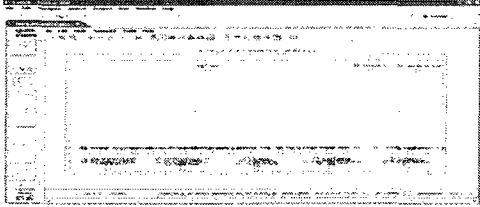
Comments on additional tasks that you may think are not added in the list and are also supported by software visualization tools for software maintenance purposes.

Appendix C. Proposed Visualization Patterns

<i>Title</i>	Radial tree
<i>Context</i>	The display consists of a number of software objects (packages  , classes  , and interfaces ) , and their inter-relationships or structural dependencies in the source code.
<i>Problem</i>	How to display large hierarchical tree structures showing dependencies among software objects?
<i>Forces</i>	<ul style="list-style-type: none"> ▪ To visualize and navigate large trees in a radial space ▪ Get an overview of the entire collection of data ▪ Zoom in on a particular item while keeping in view of the neighbourhood context ▪ Filter out uninteresting or unwanted items ▪ Get details on demand by simply clicking on an item to get a pop-up window which shows the values of each of its attributes ▪ View relationships among software objects directly ▪ Get history of the actions performed with visualization ▪ Extract sub-collections of items based on query parameters
<i>Solution</i>	<p>Use a Radial Tree representation</p>  <p>To display the detailed picture of relationships between application objects and detailed map of dependencies and dependents of every package, class or interface in an application. The idea is to display different software objects and their relationships in a radial fashion, where the object nodes are placed around the circle and their relationships are shown with directed lines emanating from the source to destination node.</p>
<i>Examples</i>	Sunburst, RadViz

<i>Title</i>	Pyramid or Skeleton View
<i>Context</i>	The display consists of a number of software objects (packages, classes and interfaces), and their inter-relationships or structural dependencies in the source code are shown indirectly by highlighting the dependent items when a user clicks on an item under selection.
<i>Problem</i>	How to display large hierarchical tree structures showing dependencies among software objects?
<i>Forces</i>	<ul style="list-style-type: none"> ▪ To visualize and navigate large software structures within fixed space of a computer screen ▪ Overview of entire structure of software system in the form of pyramid of small squares ▪ Details on demand by providing a 'data tip' to access the detailed information about any object under selection ▪ View relationships among software objects in-directly ▪ Get history of the actions performed with visualization ▪ Extract sub-collections of items based on query parameters
<i>Solution</i>	<p>Use a Skeleton View</p>  <p>This layered view of the system is constructed by putting objects (class/interface/package) that do not depend on anything at the bottom of the visualization. The objects that are dependent on the lowest layer appear in the layer above, and so on. Each square corresponds to either one object (class/interface/package) or one tangle (a square with many objects that change together).</p>
<i>Examples</i>	Icicle plot

<i>Title</i>	Nested View
<i>Context</i>	The display consists of a number of software objects (packages <input type="checkbox"/> , classes <input type="checkbox"/> and interfaces <input type="checkbox"/>) and their inter-relationships or structural dependencies in the source code.
<i>Problem</i>	How to display large hierarchical tree structures showing dependencies among software objects?
<i>Forces</i>	<ul style="list-style-type: none"> ▪ To make efficient use of the available screen space, as tree structured node-link diagrams can grow too large to be useful ▪ Overview of entire structure of software system in the form of nested rectangles ▪ Zoom in on a particular item while keeping in view of the neighbourhood context ▪ Filter out uninteresting or unwanted items ▪ Details on demand by clicking on any object under selection ▪ View relationships among software objects directly ▪ Extract sub-collections of items based on query parameters
<i>Solution</i>	<p>Use a Nested View</p>  <p>A space filling approach of visualizing large hierarchical data sets (packages, classes and interfaces) and their inter-relationships or structural dependencies in the source code. It allows to visualize the hierarchical structure (tree) by representing (mapping) the nodes with nested rectangles. The relationships among nodes are shown directly with directed lines emanating from source to destination node.</p>
<i>Examples</i>	PhotoMesa Image Browser, SmartMoney, HoneyComb, NewsMap

<i>Title</i>	Tree
<i>Context</i>	The display consists of a number of software objects (packages <input type="checkbox"/> , classes <input type="checkbox"/> and interfaces <input type="checkbox"/>) and their inter-relationships or structural dependencies in the source code.
<i>Problem</i>	How to display large hierarchical tree structures showing dependencies among software objects?
<i>Forces</i>	<ul style="list-style-type: none"> ▪ To visualize and navigate small software structures within fixed space of a computer screen ▪ Overview of entire structure of software system in the form of nodes and links ▪ Zoom in on a particular item while keeping in view of the neighbourhood context ▪ Filter out uninteresting or unwanted items ▪ Details on demand by clicking on any object under selection ▪ View relationships among software objects directly ▪ Extract sub-collections of items based on query parameters
<i>Solution</i>	<p>Use a Tree View</p>  <p>It displays the hierarchical structure (tree) by representing an acyclic graph of a set of nodes and their relationships. Nodes may represent software objects like packages, classes, interfaces and so on. Edges may represent semantic relationships among those software objects. In this structure, each element may be logically followed by two or more other elements, there is one element with no predecessor called the root node, every other element has a predecessor, and there are no circular lists.</p>
<i>Examples</i>	Visualize it!

Appendix D. Participant Evaluation Form

Thank you for completing this form based on your own background and experiences. All responses will remain anonymous and results will be used solely for research purposes.

D.1 Participant's Profile

1. Age: _____ years
2. Gender: Male Female
3. What is your first language?
 English French Other (Please specify): _____
4. What is your field of study?

5. What is your highest level of education?

6. What is your current position/employment?
 Professor Student Employee
 Other (Please specify): _____
7. Are you a left-handed person? Yes No
8. Do you wear glasses? Yes No
9. Color Blinded-ness i.e. difficulty in distinguishing certain colors?
 No Yes (Please specify): _____

D.2 Software Visualization and Maintenance Knowledge

1. How many graduate-level software maintenance courses have you taken?

None 1-2 >2 courses

2. How would you rate your knowledge of the Software Maintenance?

None Basic Intermediate Advanced

3. Have you ever used any Visualization tool?

No Yes If yes, which one(s) and for what purpose?

4. How would you rate your experience with Software Visualization tools (i.e. Creole and Structural Analysis for Java (SA4J))?

None Basic Intermediate Advanced

D.3 Application Experience

1. How would you rate your experience with Java language?

None Basic Intermediate Advanced

D.4 Hobbies and Interests

1. Please list some of your hobbies and interests below:

Thank You!! 😊

Appendix E. Informed Consent to Participate in Research



This is to state that I agree to participate in a research study on *Developing a Framework to Measure Comprehension Support Provided by Visualization Systems* conducted at the Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada. The main researchers are: Dr. Ahmed Seffah, Dr. Sudhir Mudur and Harkirat Kaur Padda. Harkirat Padda, a PhD candidate (padda@encs.concordia.ca, phone 514-848-2424 ext. 7165) is in charge of the study.

E.1 Purpose of The Study

I have been informed that the purpose of the research is to evaluate two software visualization tools, Structural Analysis for Java (SA4J) and Creole. These tools visualize the static structure of a Java software system depicting the dependencies among various software objects using the underlying visualization techniques. The tool SA4J uses two techniques, i.e. ‘radial visualization’ and ‘pyramid or skeleton visualization’, to show the static structure of a Java program. Creole uses techniques like ‘treemap’ and ‘tree’ visualization to represent the structure of a Java software system. For the comprehension study, these four visualization techniques will be considered.

E.2 Procedure

The study is designed to ask software professionals, having knowledge and experience in the field of software maintenance and visualizations in general, to explore the two software visualization tools – SA4J and Creole. You will be asked to answer a questionnaire comprising of 23 multiple choice questions for each of the 4 visualization techniques, and to accomplish 1 simple task using these visualization tools. This should take no more than 120 minutes of your time.

For the purposes of the study, an audio and video recording of your interactions with the tools will be stored in the database.

We anticipate no risk to you as a result of your participation in this study other than the inconvenience of the time to complete the questionnaire and to use the visualization tools. While there may be no immediate benefit to you as a result of your participation in this study, it is hoped that we may gain valuable information about your experiences to develop a comprehension measurement framework.

All provided information and data collection will be stored anonymously in a database, with no information that can identify participants. Results from this study will be reported only in aggregate form in scientific communications like articles, workshops, and conference presentations. A complete summary of the results will be published in a thesis.

If you wish to receive the results of this study, please check the box below and provide an e-mail address.

Name:

Mail address:

E.3 Conditions of Participation

- I understand that I am free to withdraw my consent and discontinue my participation at anytime without negative consequences.
- I understand that my participation in this study is CONFIDENTIAL (i.e., the researcher will know, but will not disclose my identity)
- I understand that the data from this study may be published.

I HAVE CAREFULLY STUDIED THE ABOVE AND UNDERSTAND THIS AGREEMENT. I FREELY CONSENT AND VOLUNTARILY AGREE TO PARTICIPATE IN THIS STUDY.

NAME (please print) _____

SIGNATURE _____

If at any time you have questions about your rights as a research participant, please contact Ms. Adela Reid, Research Ethics and Compliance Officer, Concordia University, at (514) 848-7481 or by email at areid@alcor.concordia.ca or Adela.Reid@concordia.ca

Appendix F. Checklist for Study

F.1 Before the Test Begins

1. The following documents should be ready:
 - a. Consent form
 - b. Pre-test questionnaire
 - c. Task document
 - d. Visualization patterns documents
 - e. Post-test questionnaires
2. Tools should be running and software program should be parsed.
3. Webcam should be adjusted for sharp image.
4. Both the microphone sets should be adjusted to same 'RF channel' and should be powered on throughout the recording.
5. Morae Recorder settings should be set as follows:
 - a. File name should be given for saving the recording
 - b. Check that the folder for saving the recording is not changed
 - c. Capture options should be ticked for microphone, keystrokes, screen text and mouse clicks
 - d. Ensure that the 'mouse clicks' options should be ticked for highlighting the effects of left mouse clicks, middle mouse clicks, and right mouse clicks
 - e. Visibility during recording should be set to 'minimize to tray'
 - f. 'Start details' and 'Stop details' should be adjusted to manual setting
 - g. The Settings option under the Record menu should be adjusted for 'Lossless video' and the audio of the wireless device should be set to maximum

6. For SA4J: Make sure the UI skin is adjusted to 'Structural Analysis'
 7. For Creole: Make sure the Node Labels are adjusted to Above Node (fixed) and Navigation to 'Fisheye'
 8. Perform a test recording to make sure that recording will be successful and is audible later on.
-

F.2 During the Test

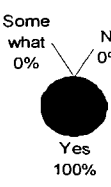
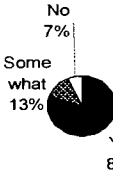
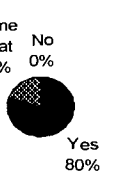
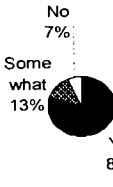
1. Welcome the participant
 2. Get consent form signed
 3. Get the pre-test questionnaire filled in
 4. Give visualization patterns for the coaching session
 5. Give the task document and the questionnaires
 6. Persuade them to think-aloud while performing the assigned task
-

F.3 After the Test

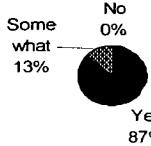


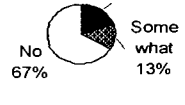
1. Ask a final question: rank the four visualization techniques in order of your likeability to depict the static structure of underlying software system?
2. Ask the participant for signatures after giving a thanking gift.

Appendix G. Analysis of Participants' Responses

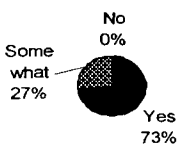
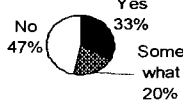
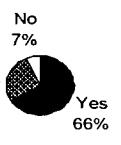
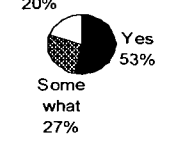
1. Are you able to navigate from one location to another in the visualization?

Radial	Pyramid	NestedView	Tree
 <p>Some what 0% No 0% Yes 100%</p>	 <p>No 7% Some what 13% Yes 80%</p>	 <p>Some what 20% No 0% Yes 80%</p>	 <p>No 7% Some what 13% Yes 80%</p>

2. Are you able to undo your manipulation operations (e.g. select, click, move etc.) with the visualization to go back successfully to previously displayed screen?

Radial	Pyramid	NestedView	Tree
 <p>Some what 13% No 0% Yes 87%</p>	 <p>Yes 33% No 60% Some what 7%</p>	 <p>Yes 13% No 40% Some what 47%</p>	 <p>Yes 20% No 67% Some what 13%</p>

3. Are you able to see the location of any information object with respect to an overall context of other information objects in the display?

Radial	Pyramid	NestedView	Tree
 <p>Some what 27% No 0% Yes 73%</p>	 <p>Yes 33% No 47% Some what 20%</p>	 <p>No 7% Some what 27% Yes 66%</p>	 <p>No 20% Some what 27% Yes 53%</p>

4. Does the organization of menus seem logical (i.e. are related tasks grouped together)?

Radial	Pyramid	NestedView	Tree

5. Is the visualization symmetrically well-represented (i.e. organized vertically or horizontally) to utilize the screen space?

Radial	Pyramid	NestedView	Tree

6. Is there only necessary (i.e. redundant, not reasonable) information on the screen?

Radial	Pyramid	NestedView	Tree

7. Are you able to understand the meanings of icons/symbols/labels used in the display?

Radial	Pyramid	NestedView	Tree

8. Are you able to clearly identify the information objects displayed in the visualization despite any overlapping?

Radial	Pyramid	NestedView	Tree
<p>No 7% Yes 60% Some what 33%</p>	<p>No 20% Yes 33% Some what 47%</p>	<p>No 20% Yes 47% Some what 33%</p>	<p>No 53% Yes 20% Some what 27%</p>

9. Are the used visual attributes (like - size, shape, color, texture etc.) for icons/information objects appropriate to distinguish them in display?

Radial	Pyramid	NestedView	Tree
<p>No 0% Yes 87% Some what 13%</p>	<p>No 13% Yes 0% Some what 87%</p>	<p>No 7% Yes 66% Some what 27%</p>	<p>No 0% Yes 87% Some what 13%</p>

10. Can the icons/symbols/labels be interpreted or expressed in only one way (i.e. there is no ambiguity in their meaning)?

Radial	Pyramid	NestedView	Tree
<p>No 13% Yes 20% Some what 67%</p>	<p>No 33% Yes 40% Some what 27%</p>	<p>No 7% Yes 73% Some what 20%</p>	<p>No 13% Yes 80% Some what 7%</p>

11. Are you able to see the most important element in the display based on any visual attribute like - color, motion, shape, size, texture etc.?

Radial	Pyramid	NestedView	Tree
<p>Some what 0% No 0% Yes 100%</p>	<p>No 27% Some what 13% Yes 60%</p>	<p>No 27% Some what 7% Yes 66%</p>	<p>No 13% Some what 0% Yes 87%</p>

12. Are you able to differentiate (based on visible change in shape/color/size/brightness etc.) the object that you select from the one that is not selected?

Radial	Pyramid	NestedView	Tree
<p>Some what 0% No 0% Yes 100%</p>	<p>Some what 0% No 7% Yes 93%</p>	<p>Some what 7% No 0% Yes 93%</p>	<p>Some what 20% No 0% Yes 80%</p>

13. Is it easy to figure out how to use various artifacts (e.g. buttons, links, information objects, icons and so on) in the visualization based on the given visual cues?

Radial	Pyramid	NestedView	Tree
<p>No 0% Some what 47% Yes 53%</p>	<p>No 13% Some what 40% Yes 47%</p>	<p>No 13% Some what 40% Yes 47%</p>	<p>No 33% Some what 20% Yes 47%</p>

14. Are you able to easily manipulate (e.g., select, move, click etc.) all the artifacts (e.g. buttons, links, information objects, icons and so on) in the visualization?

Radial	Pyramid	NestedView	Tree
<p>No 13% Yes 54% Some what 33%</p>	<p>No 20% Yes 67% Some what 13%</p>	<p>No 13% Yes 47% Some what 40%</p>	<p>No 0% Yes 60% Some what 40%</p>

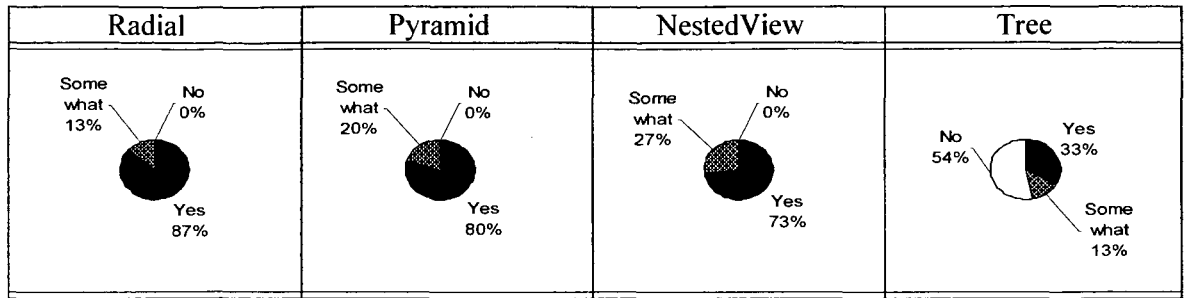
15. Just looking at the visualization, can you answer which information objects are related to one another?

Radial	Pyramid	NestedView	Tree
<p>No 0% Yes 47% Some what 53%</p>	<p>No 54% Yes 13% Some what 33%</p>	<p>No 0% Yes 47% Some what 53%</p>	<p>No 13% Yes 67% Some what 20%</p>

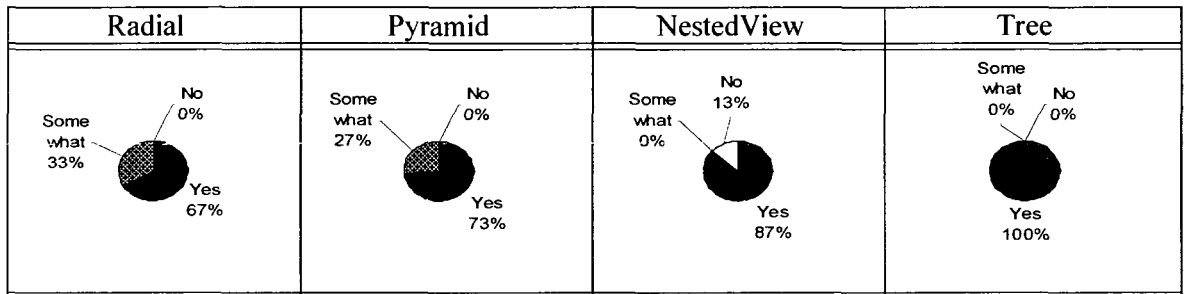
16. Do you think the visual design of the information objects reveals the features of the underlying information? (For example: size of a 'class' or depth of an information object in the information hierarchy)

Radial	Pyramid	NestedView	Tree
<p>No 46% Yes 27% Some what 27%</p>	<p>No 60% Yes 13% Some what 27%</p>	<p>No 73% Yes 7% Some what 20%</p>	<p>No 27% Yes 40% Some what 33%</p>

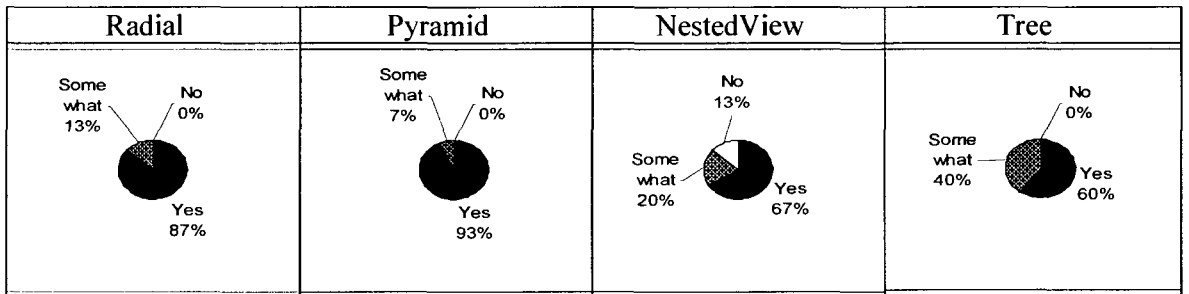
17. Are the icons/labels readable?



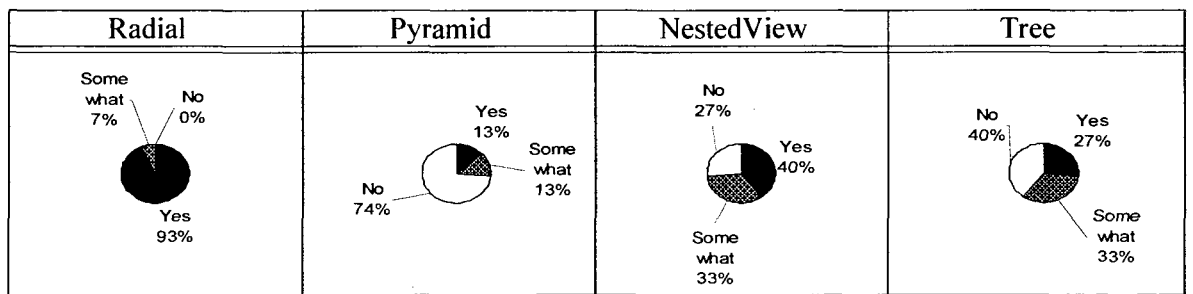
18. Is the color used for symbols/labels in good contrast to the background color?



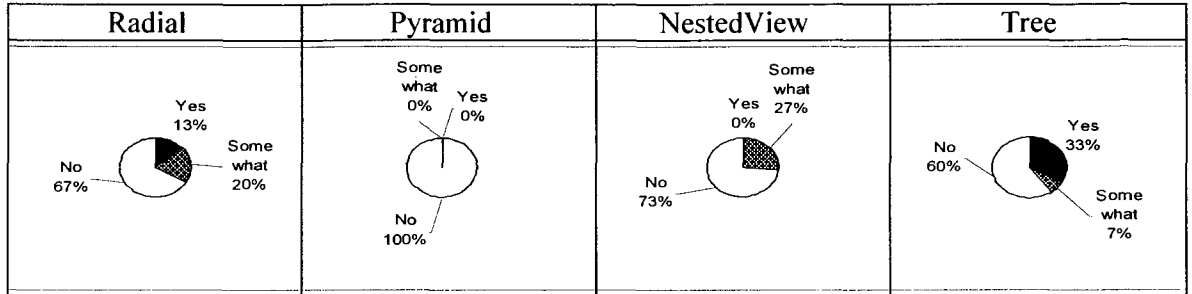
19. Is the font size and font type used for labels appropriate?



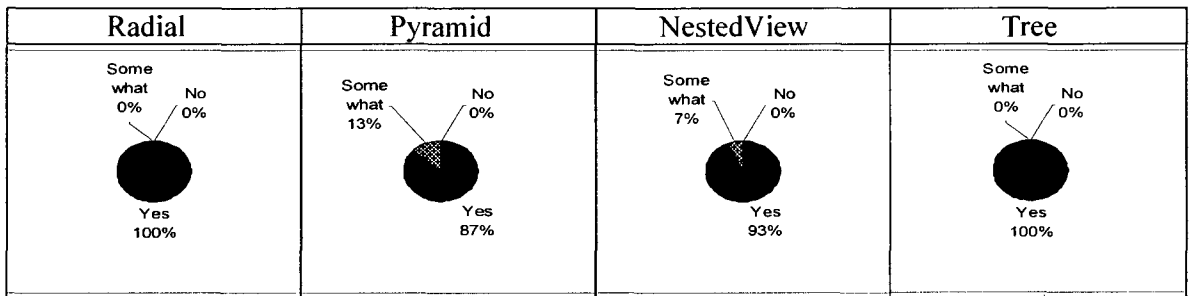
20. Do you think with this visualization you are able to effectively perform the intended tasks (Take an example: task 1 to show the related dependencies in a software system)?



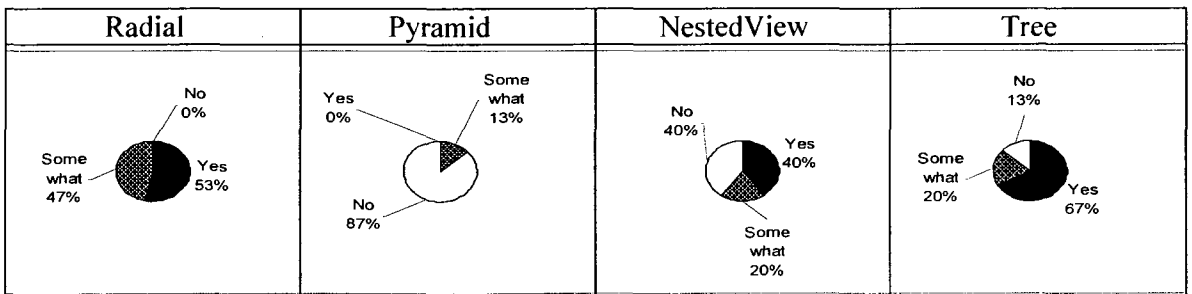
21. Do you think that there is no need of some other related views in the form of different visualizations to fully understand the underlying system?



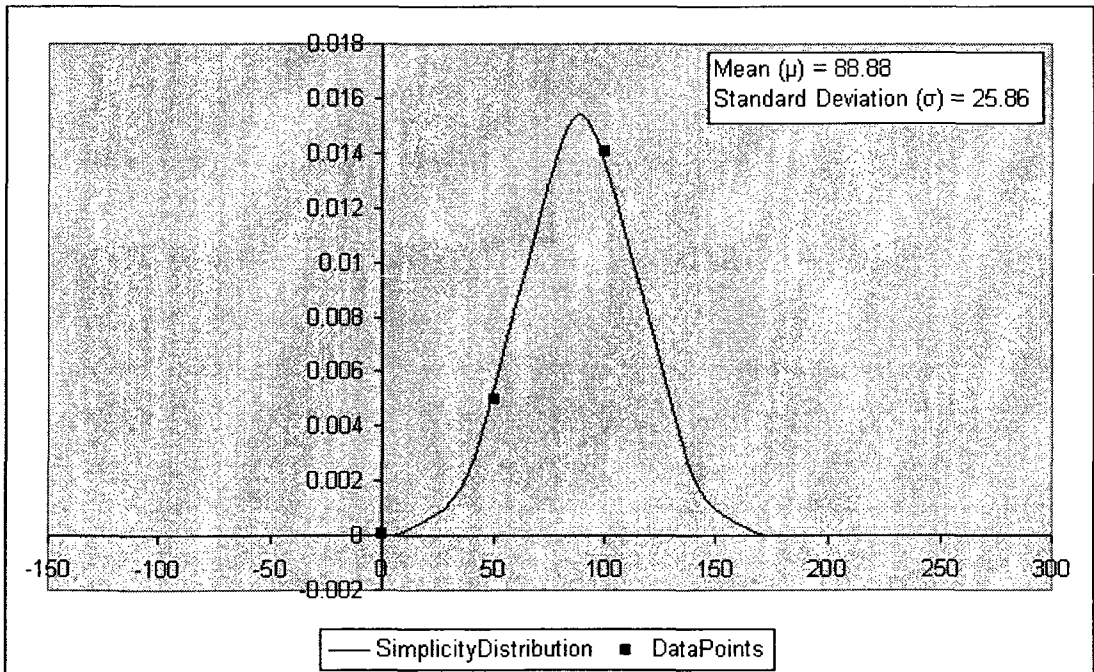
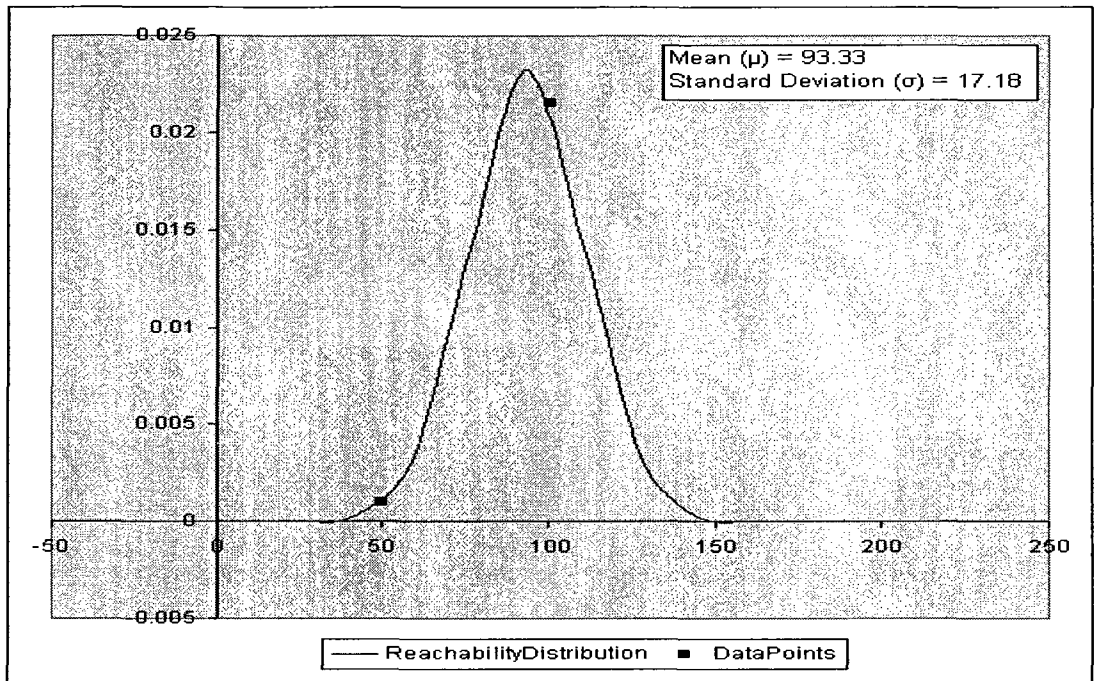
22. Are the interaction methods of input devices, like – mouse, keyboard or vests in HMD etc., natural to you? (For example: if clicking the buttons on the mouse leads you to what you want to do with the visualization system?)

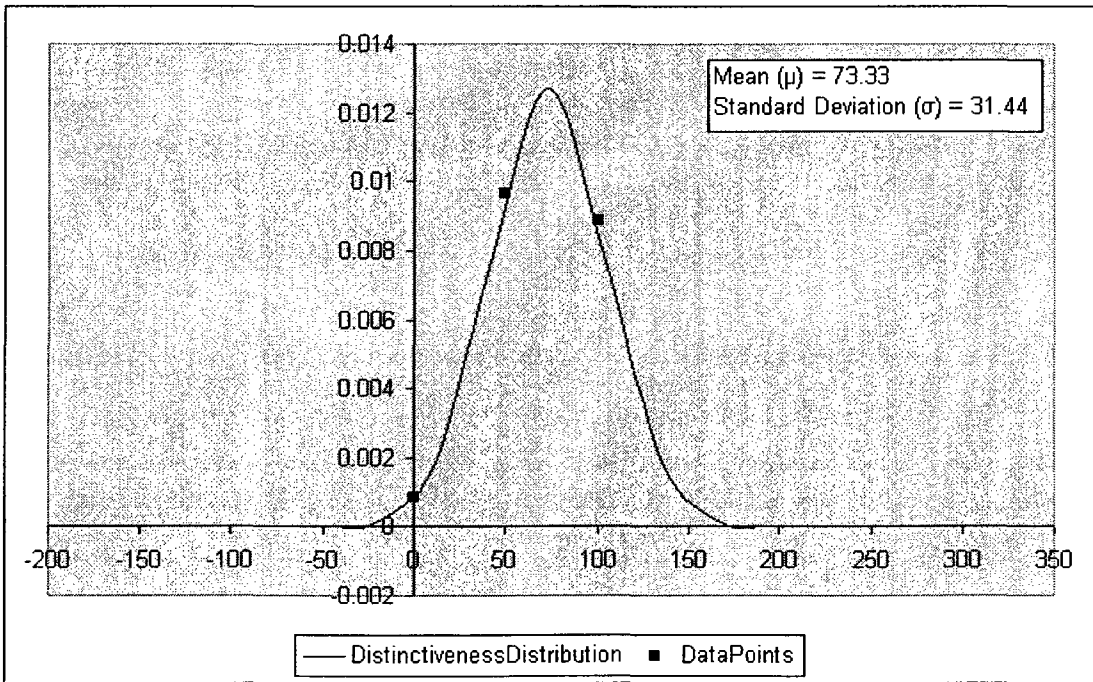
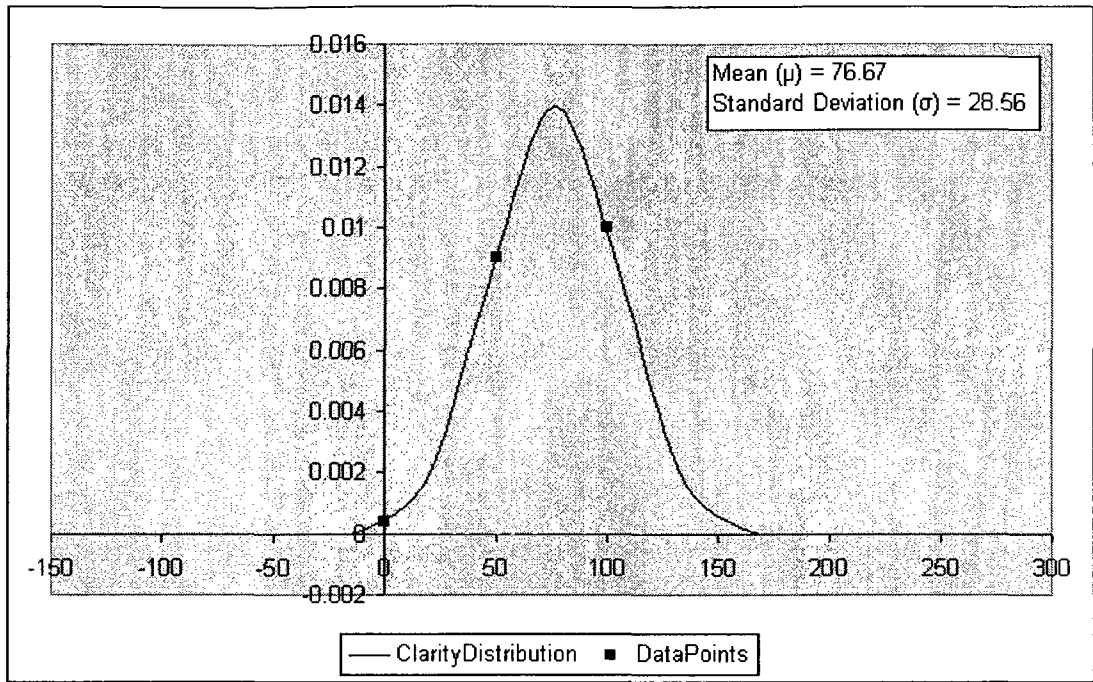


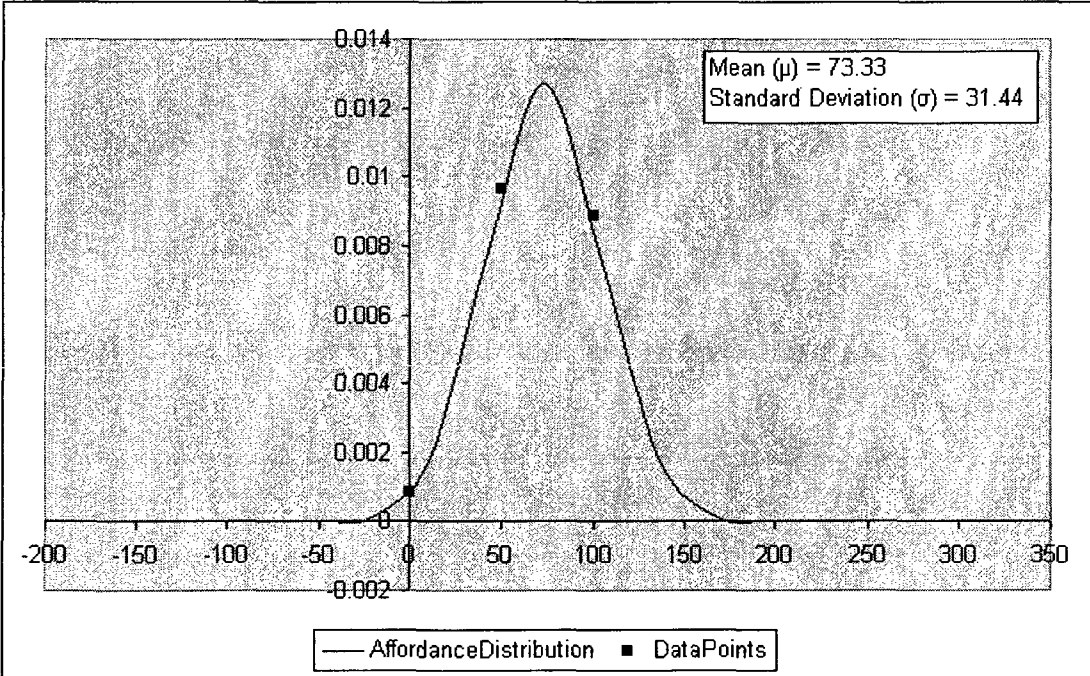
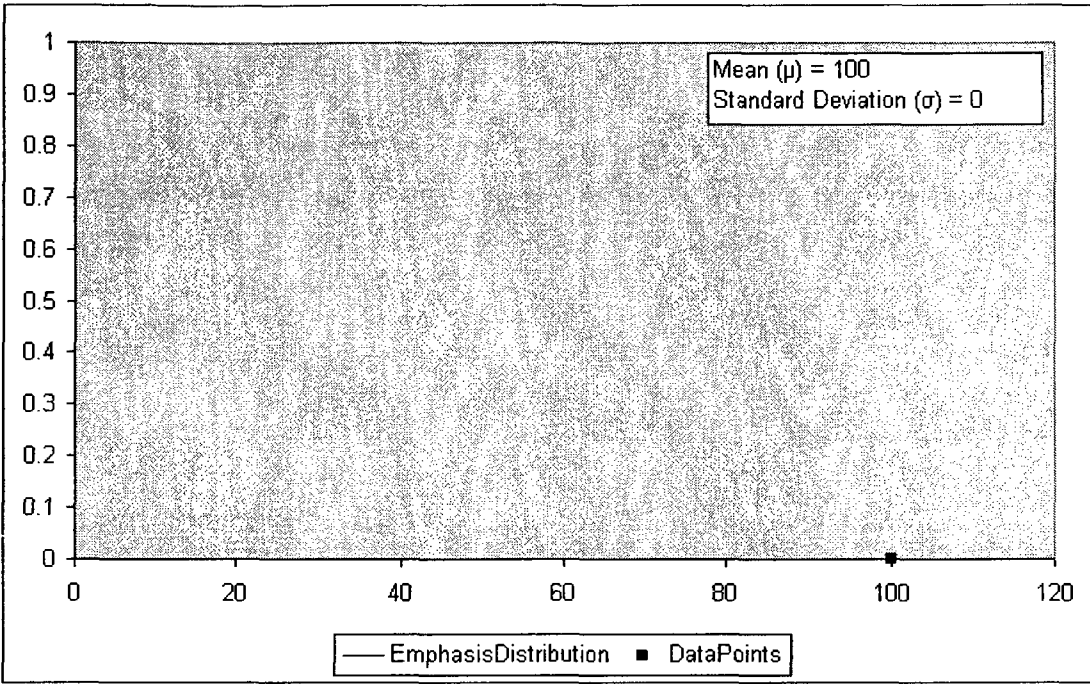
23. Are the icons/symbols used in the visualizations similar to the one that are used in the underlying domain? (For example: symbols used to depict relationships in UML.)

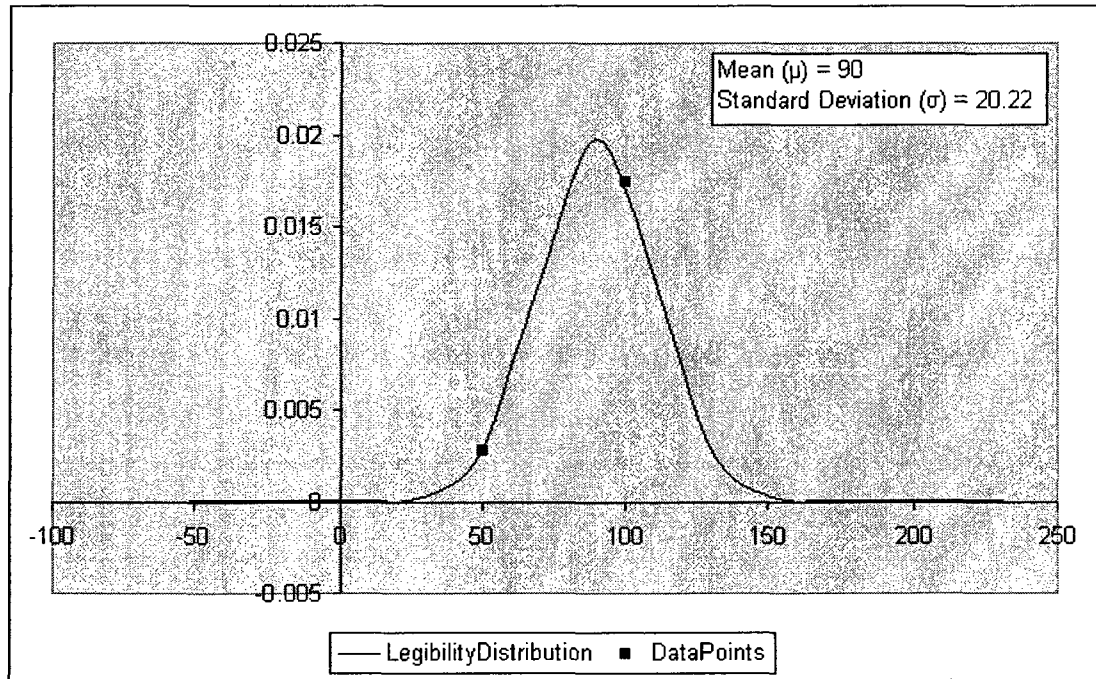
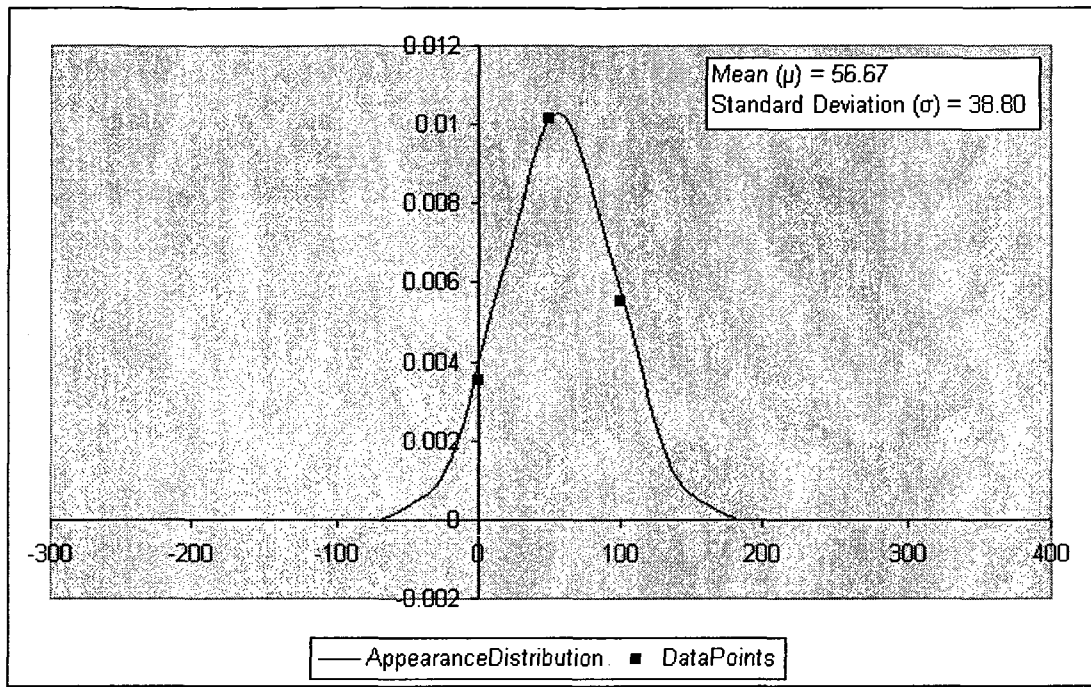


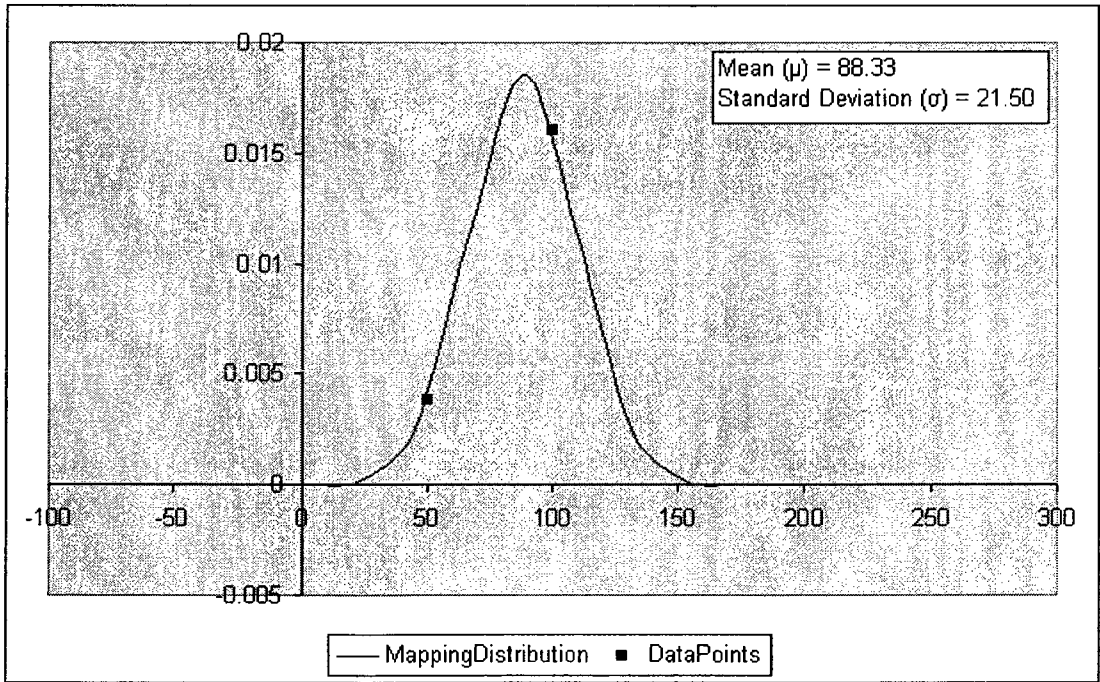
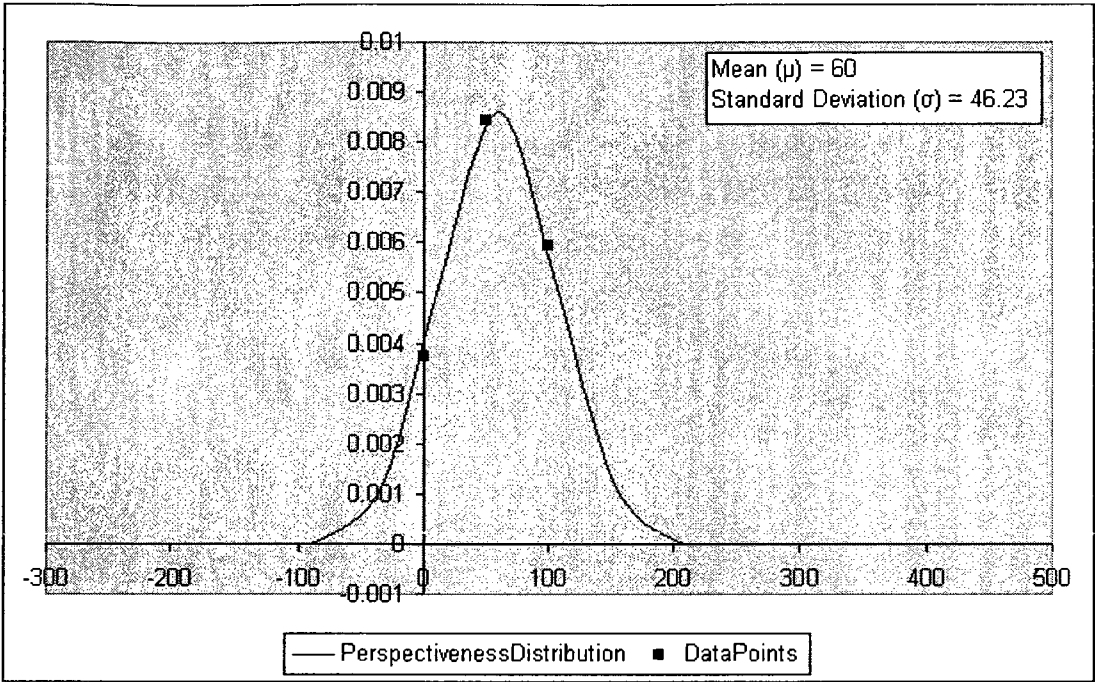
G.1 Normal Distribution Curves for Radial Technique



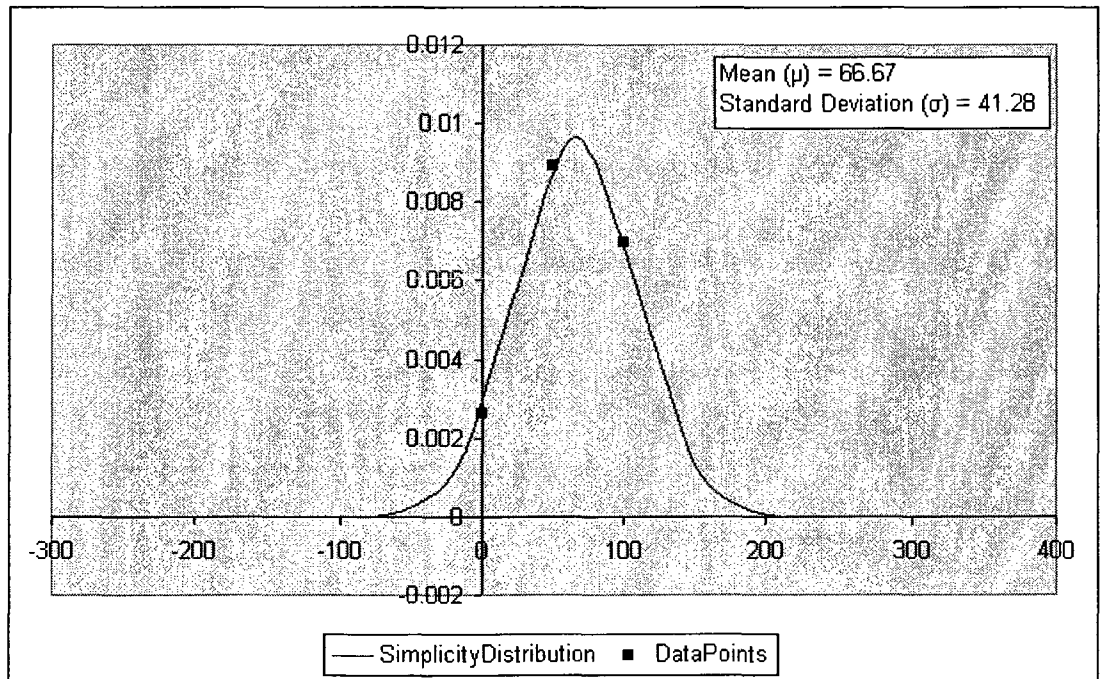
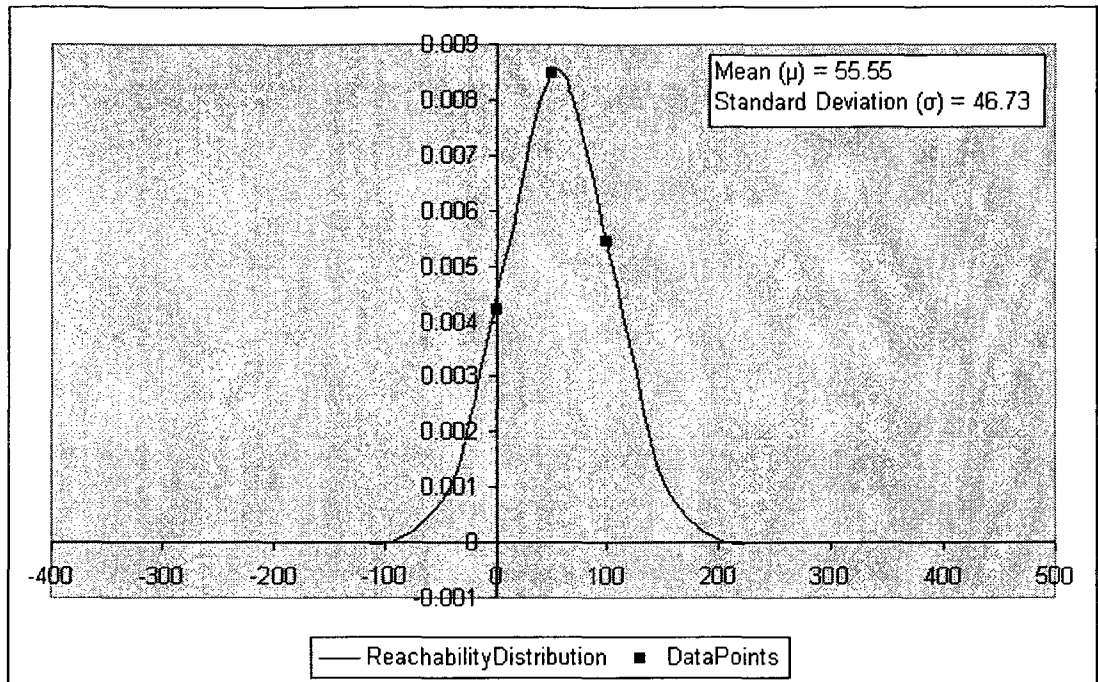


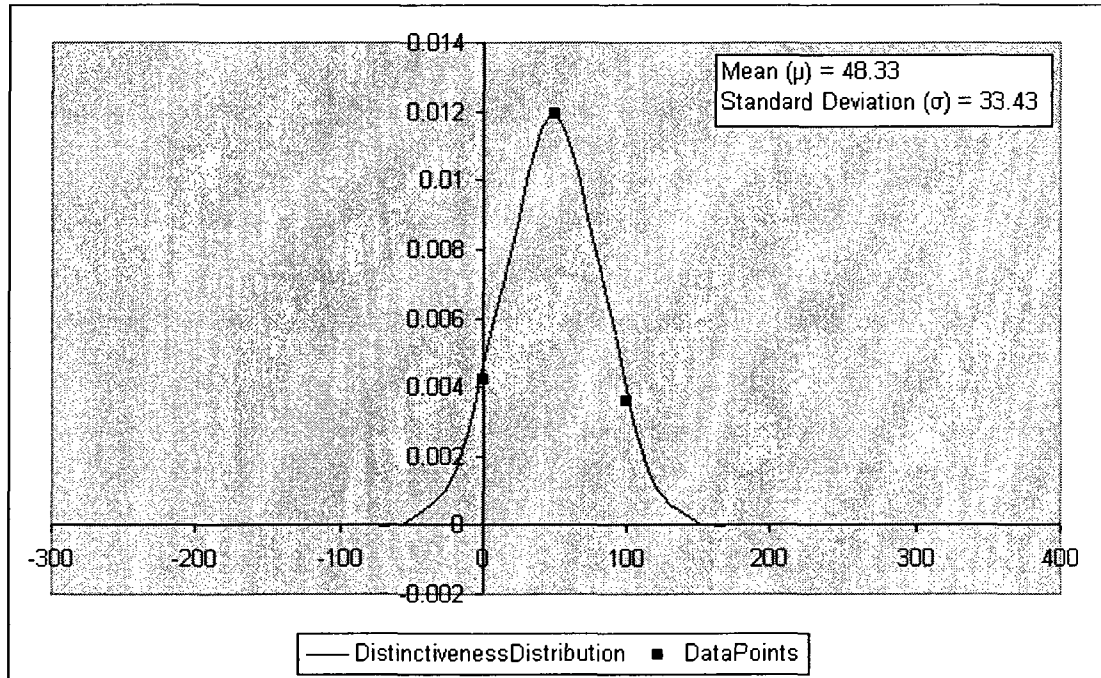
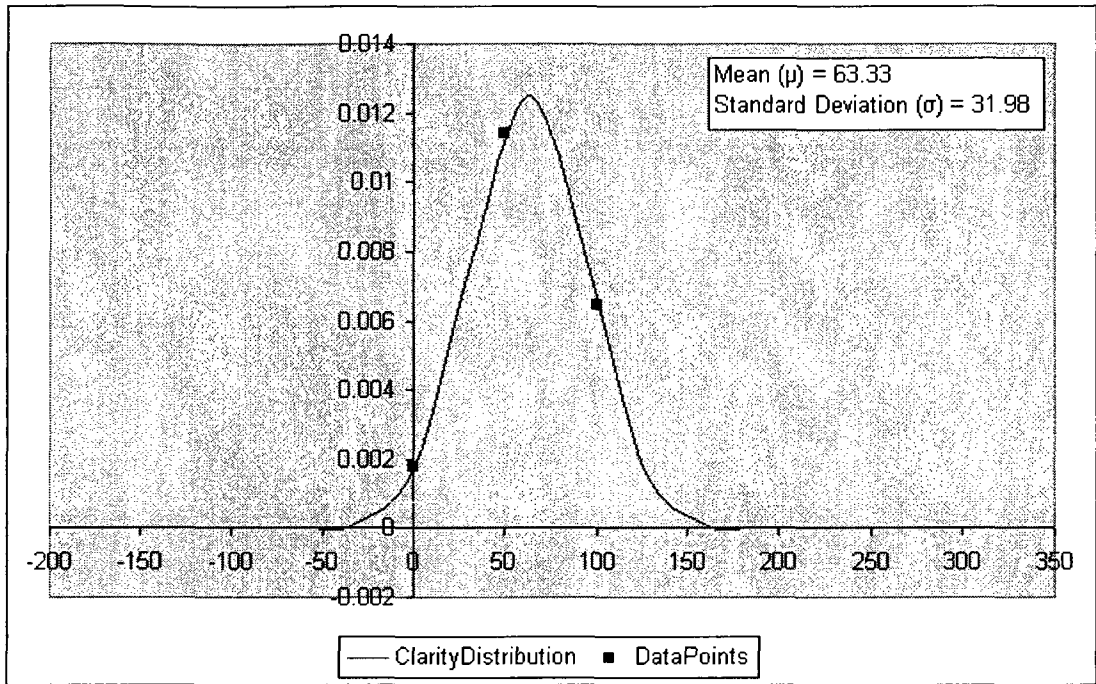


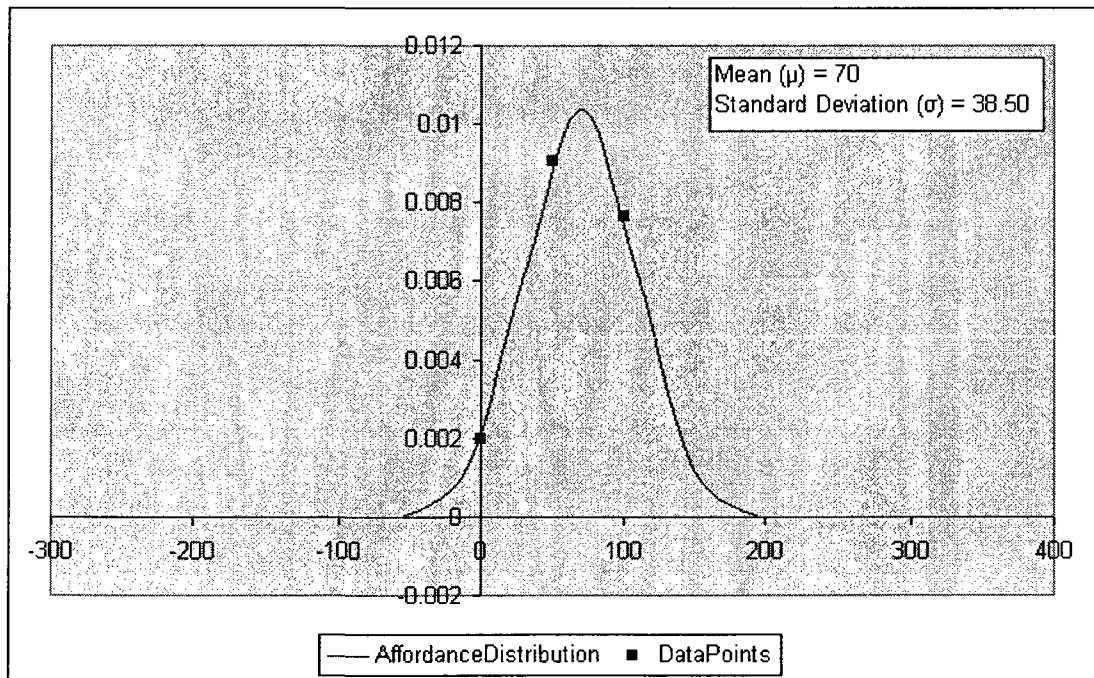
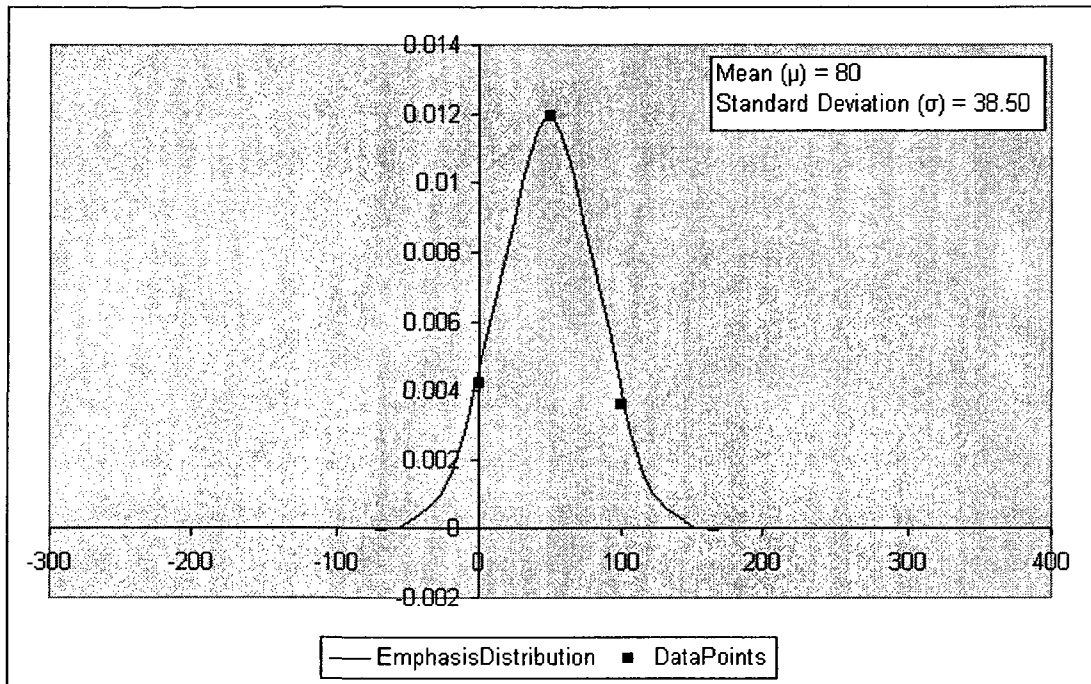


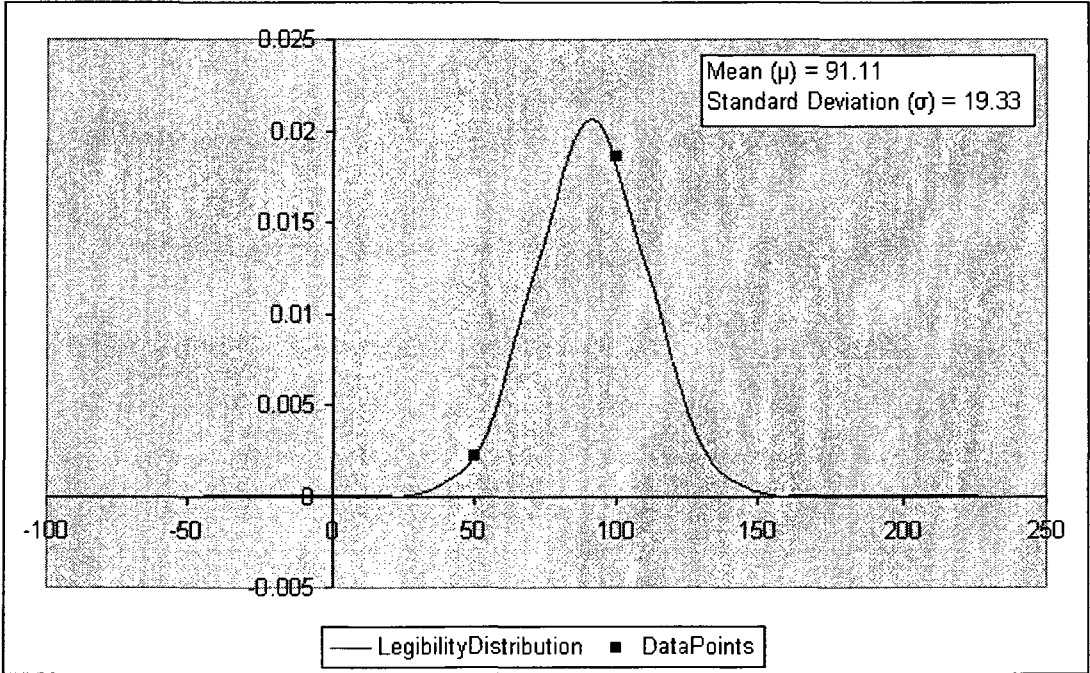
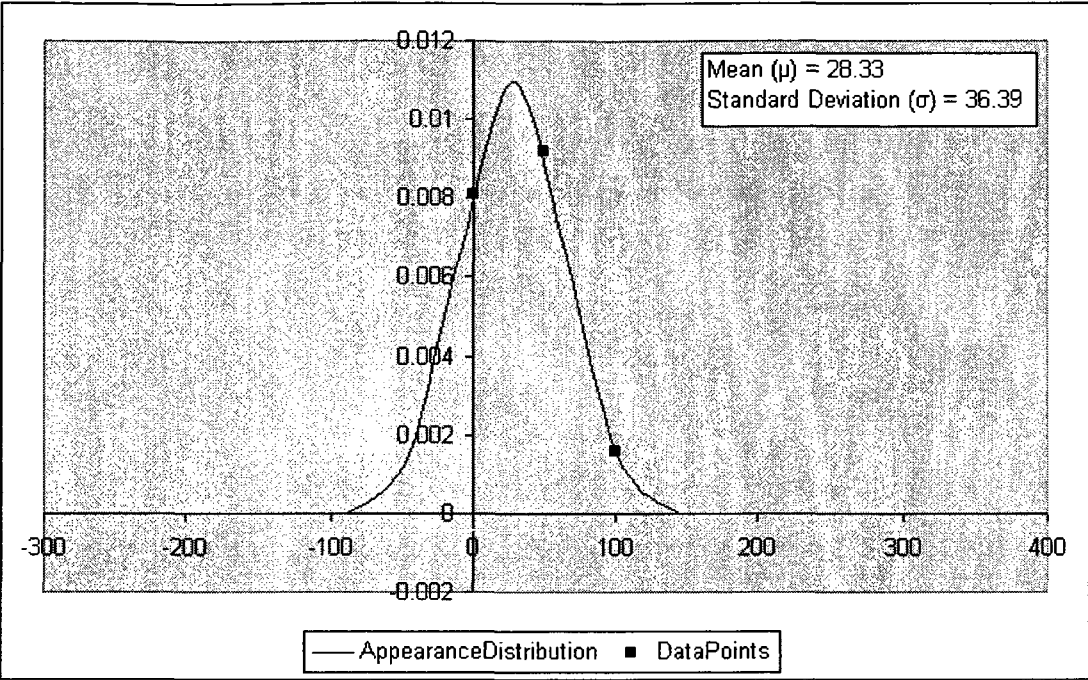


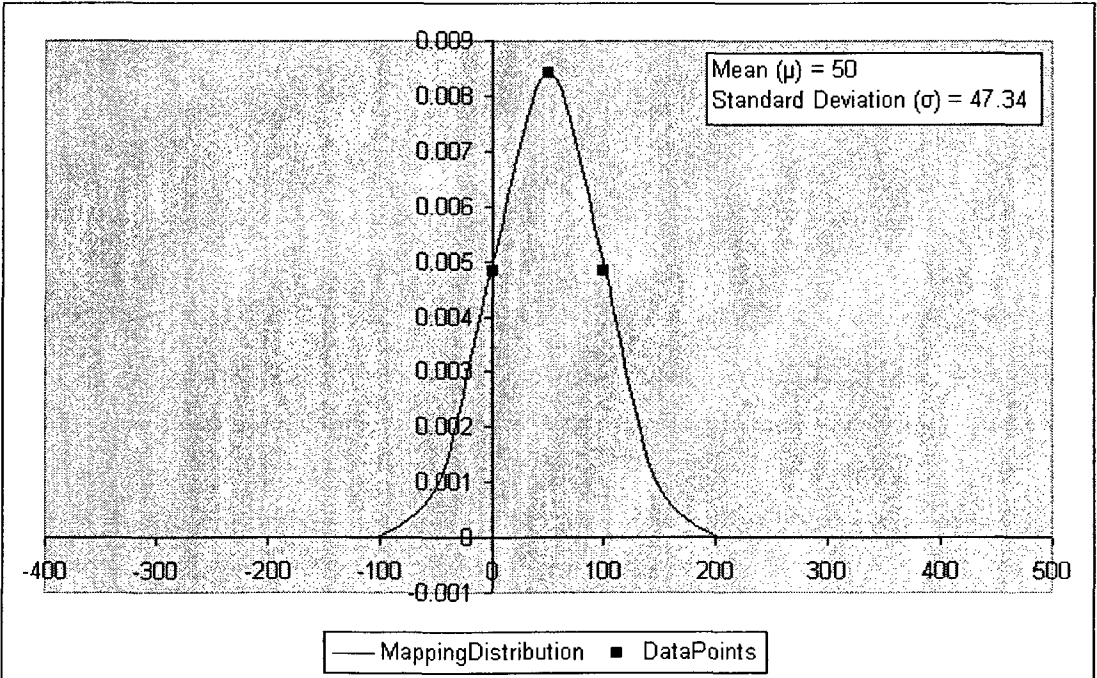
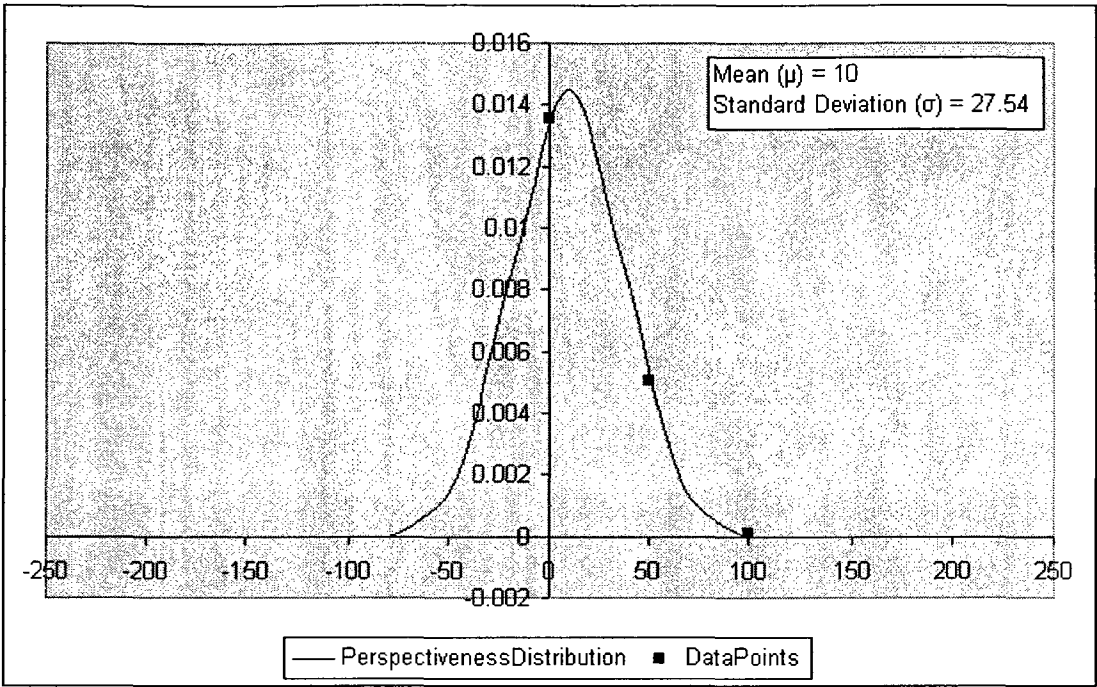
G.2 Normal Distribution Curves for Pyramid Technique



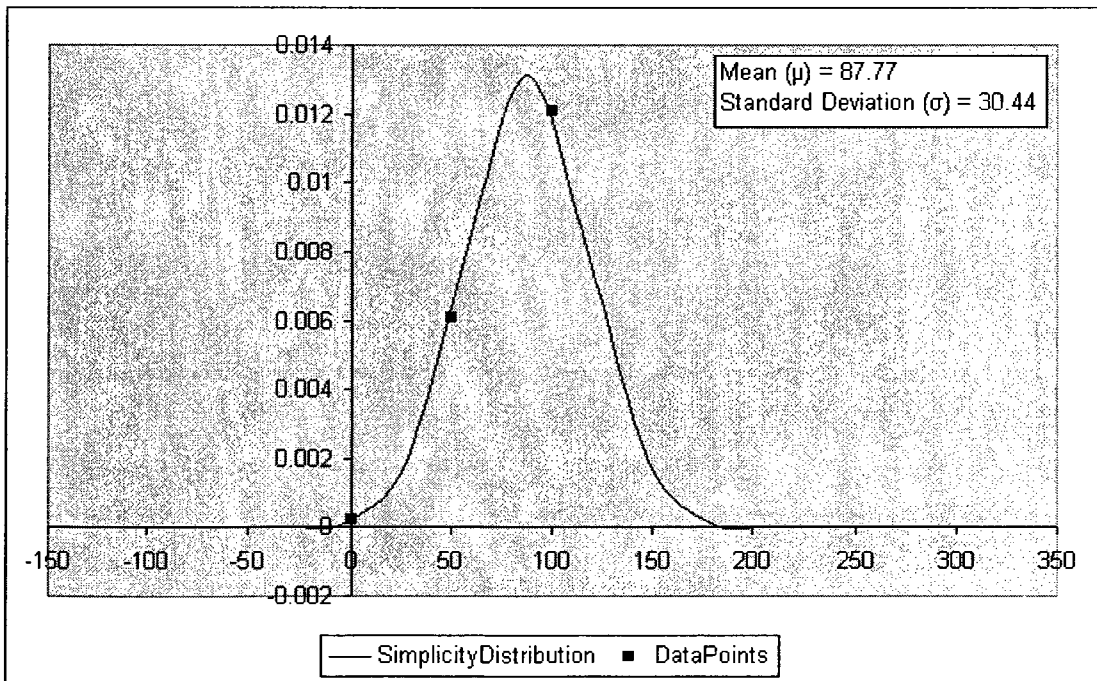
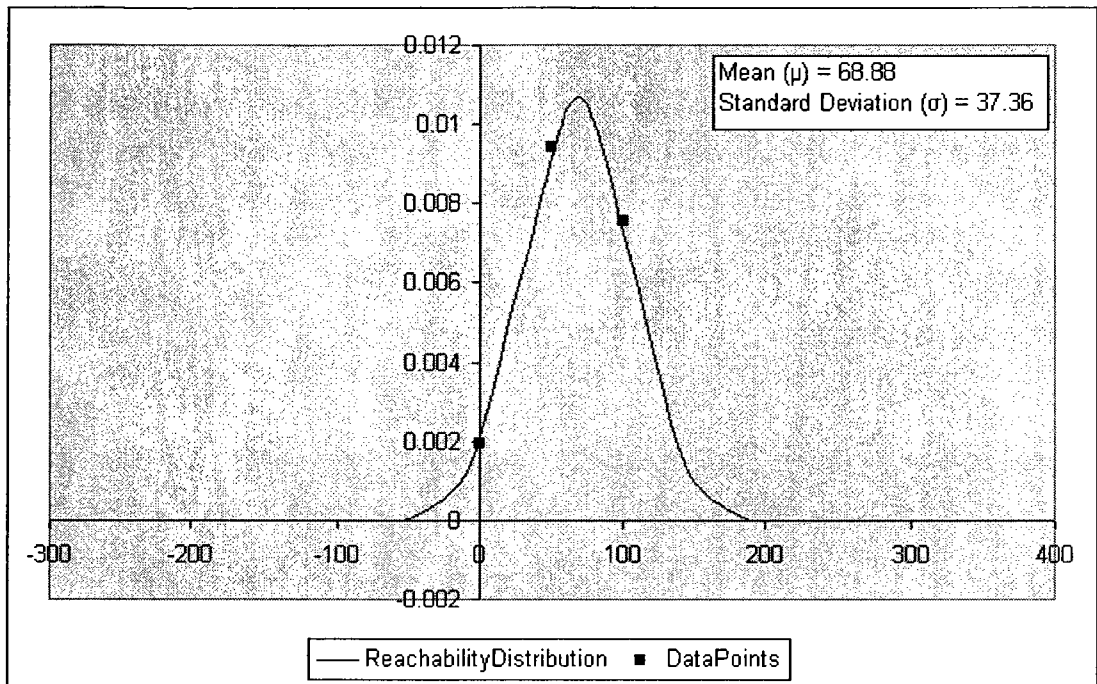


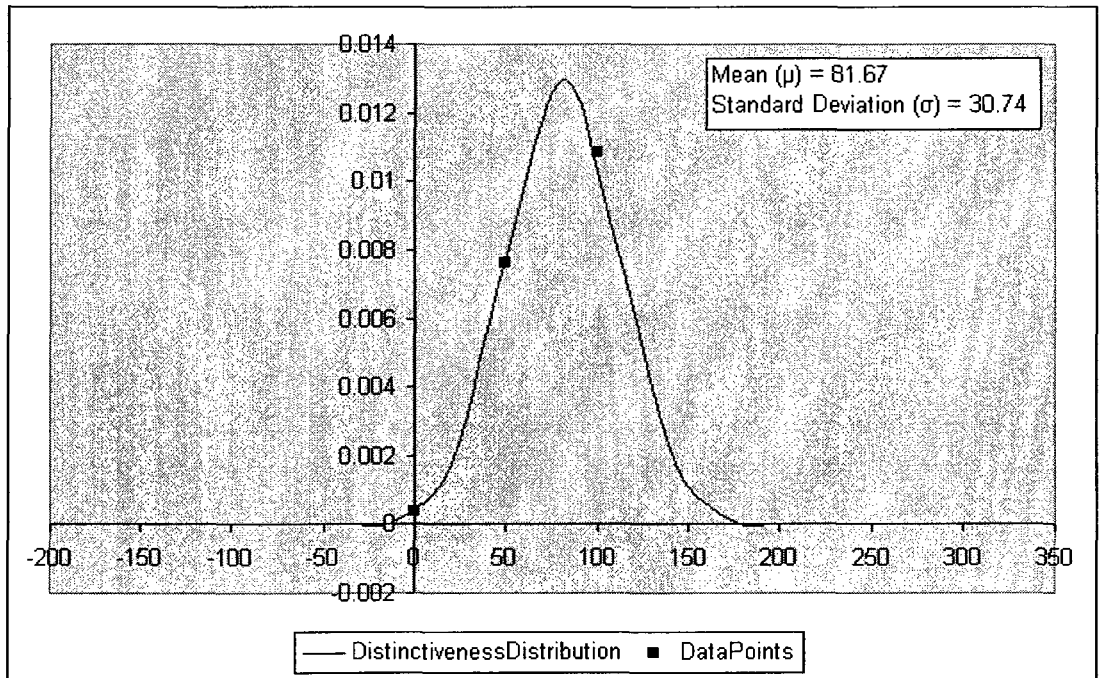
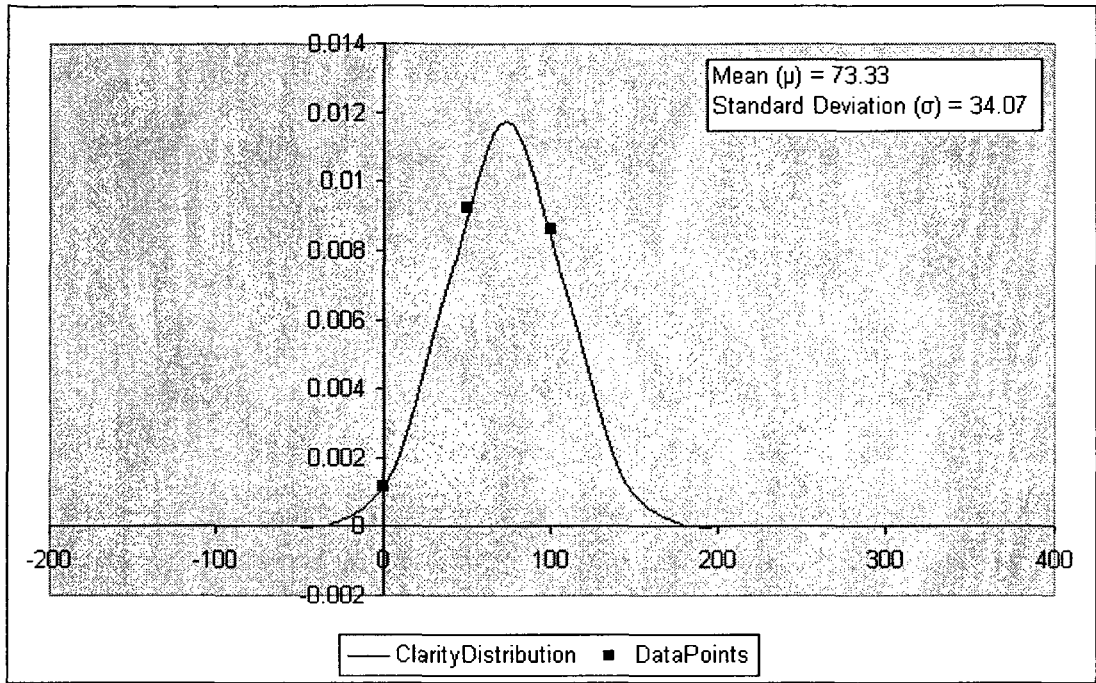


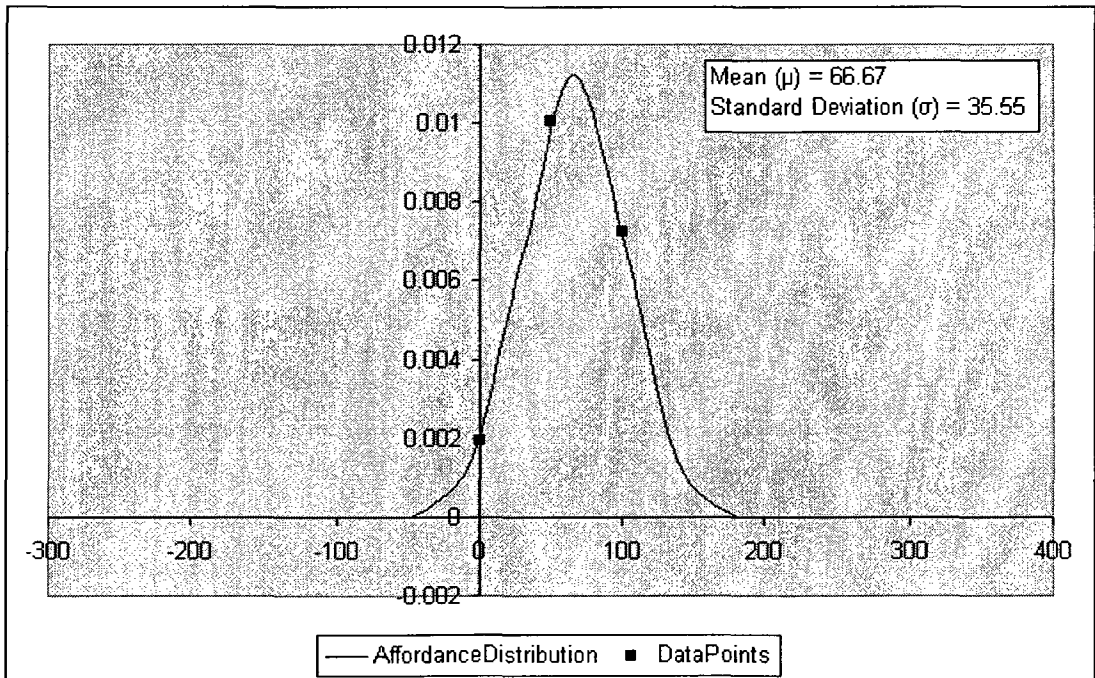
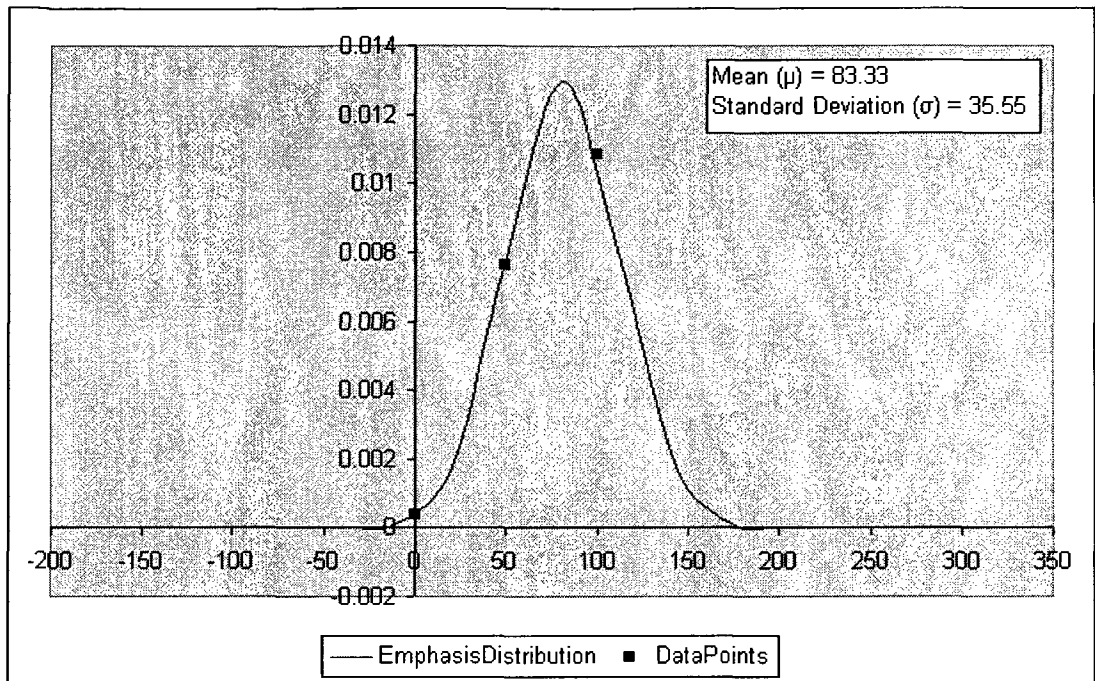


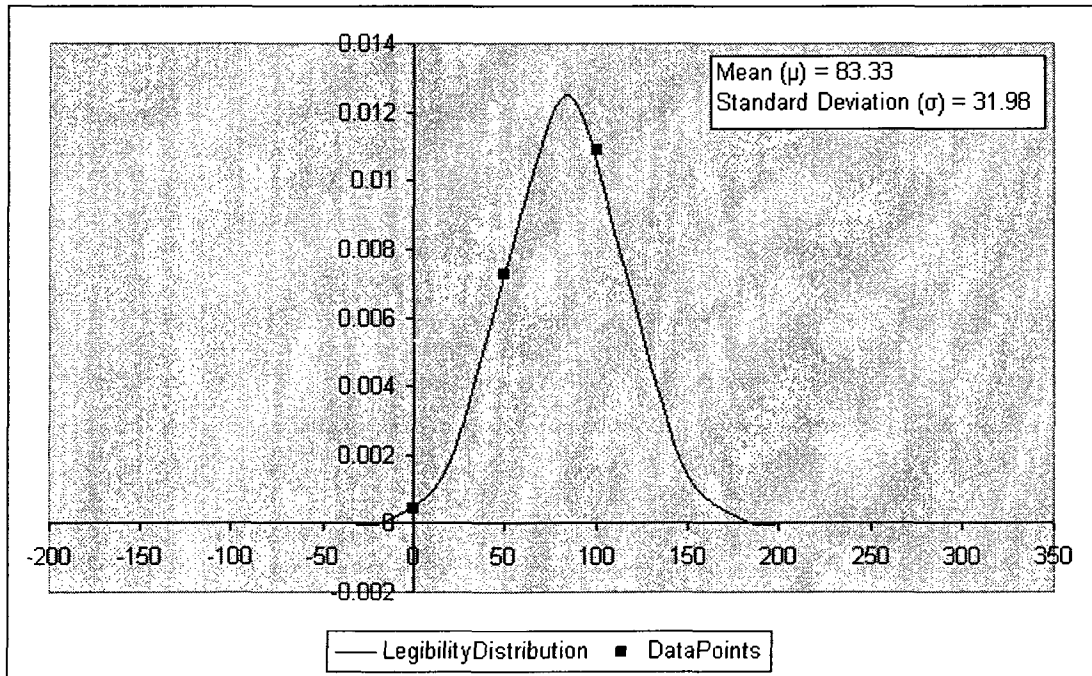
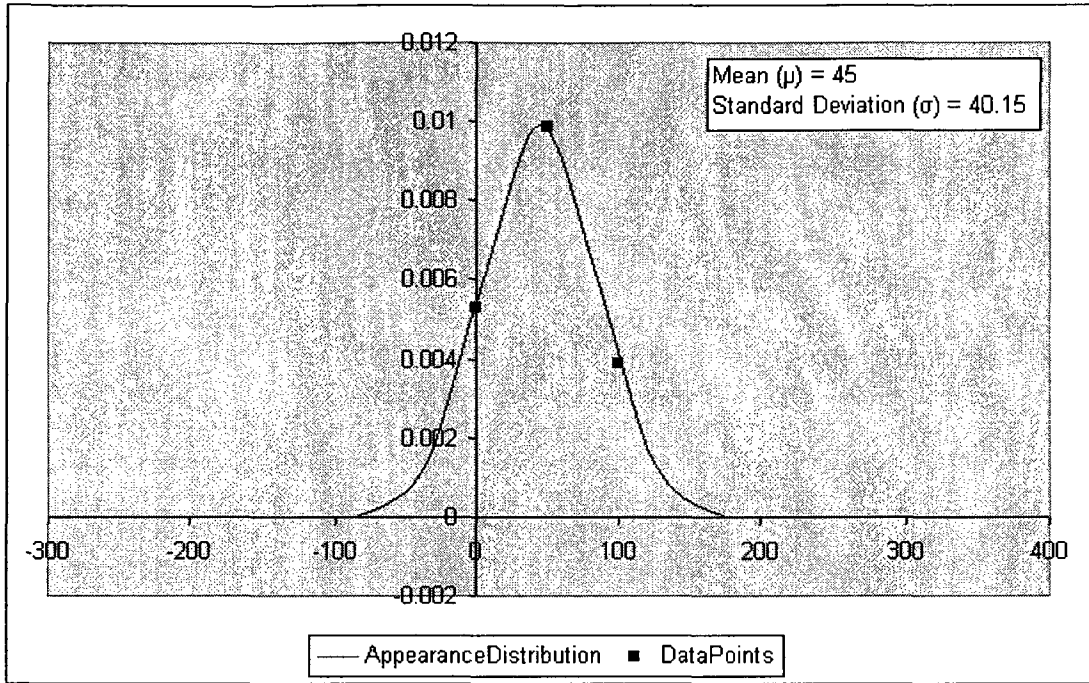


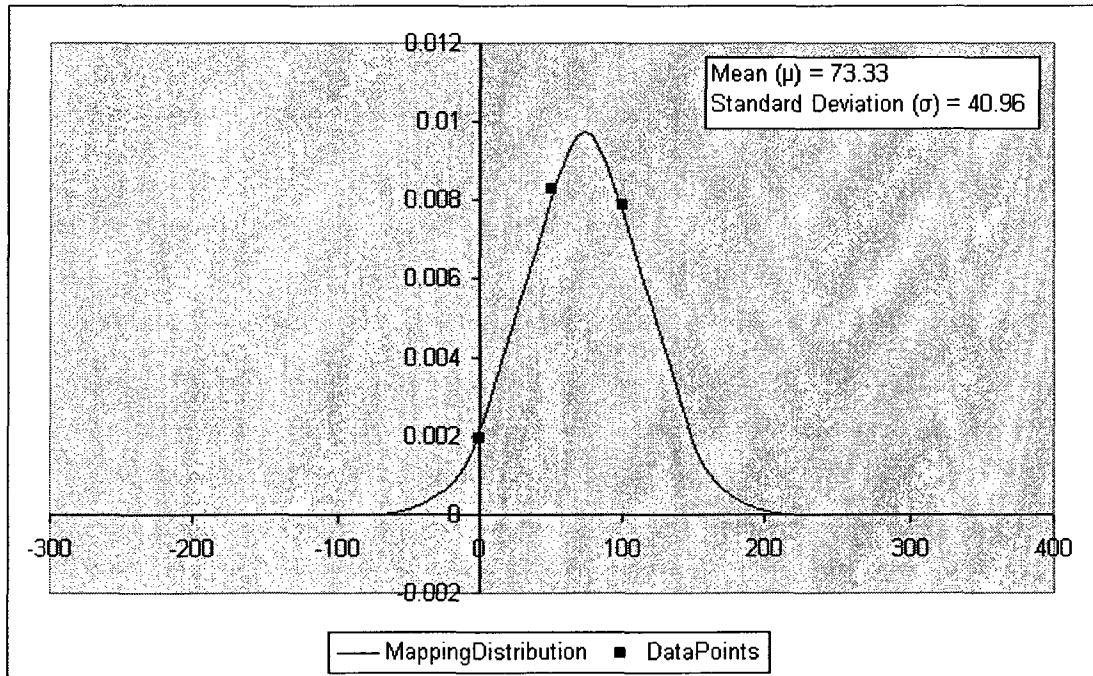
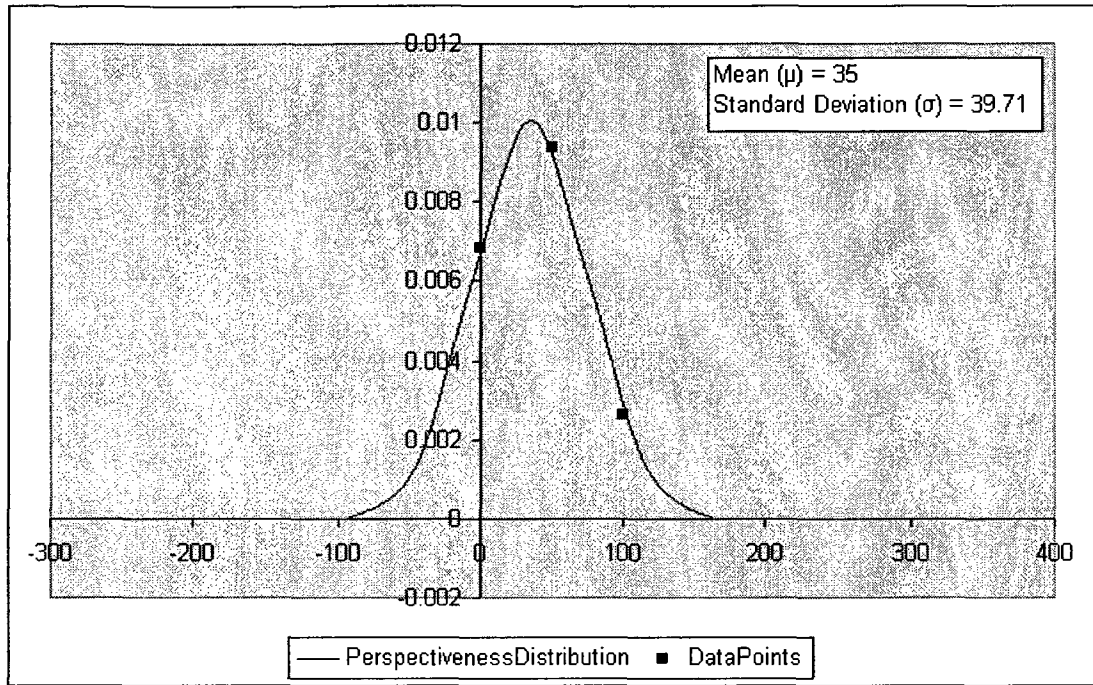
G.3 Normal Distribution Curves for NestedView Technique



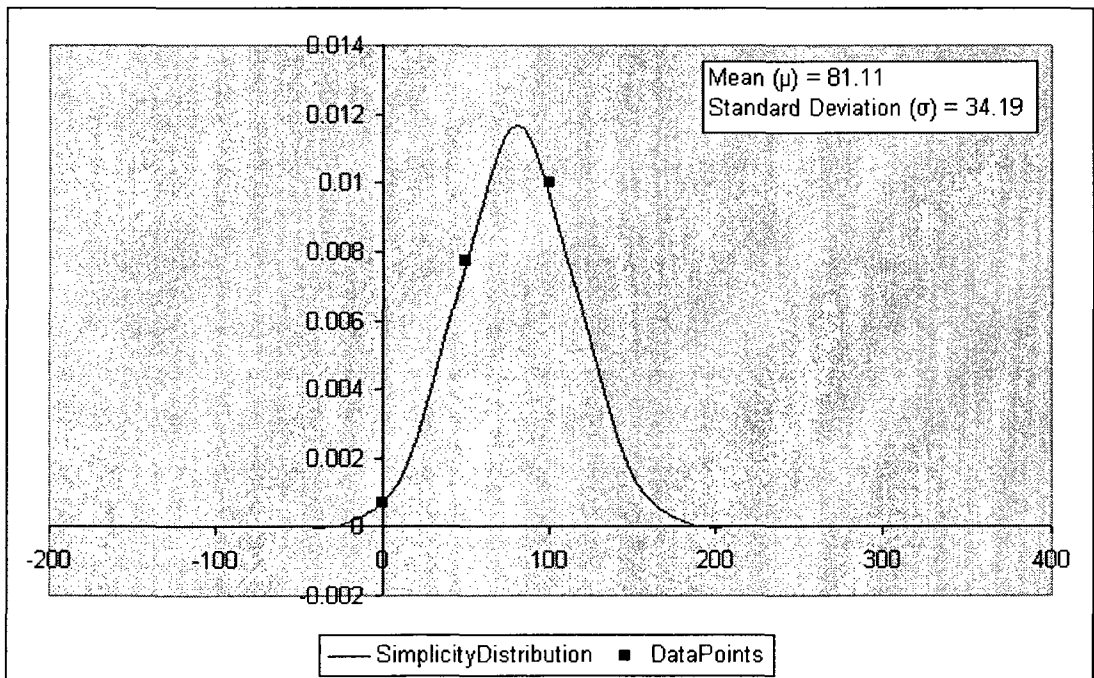
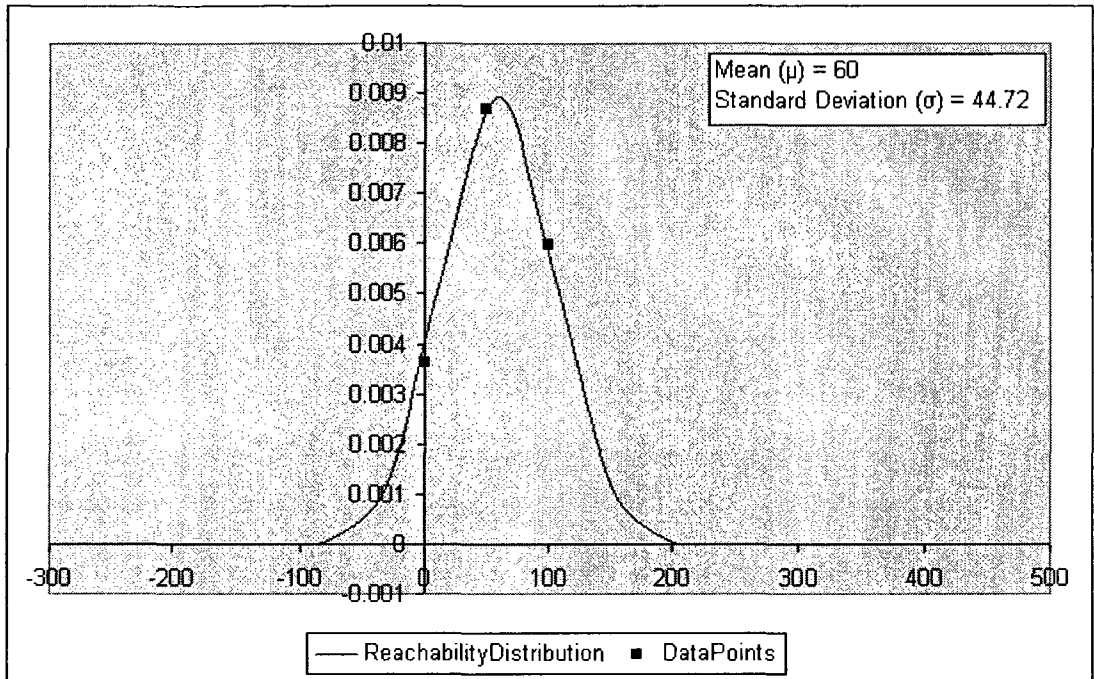


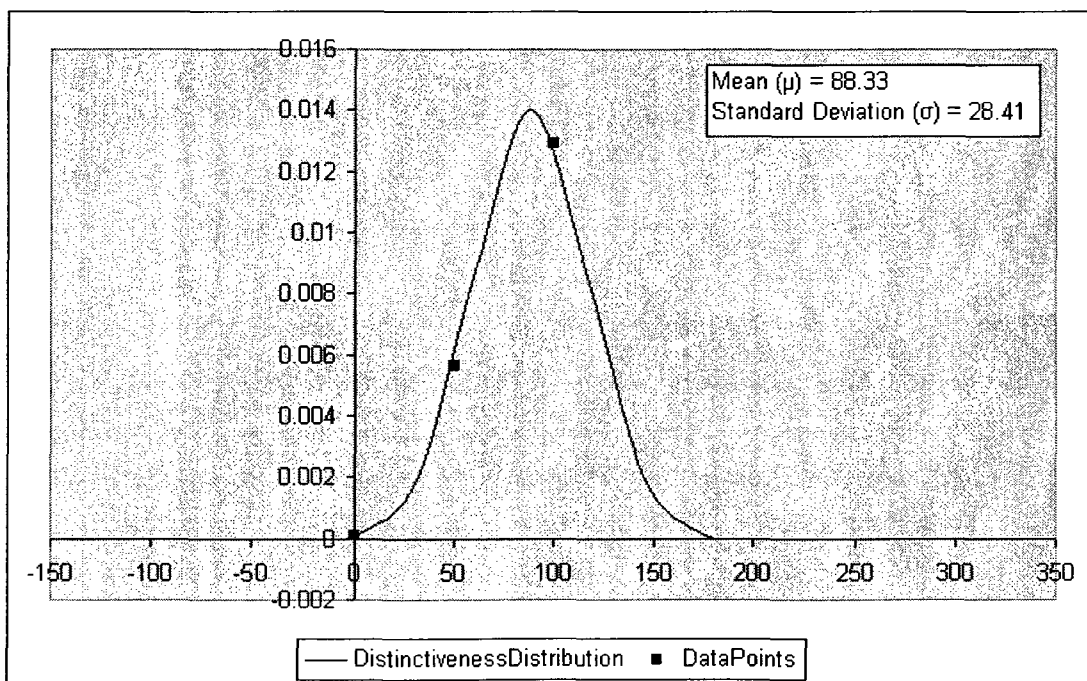
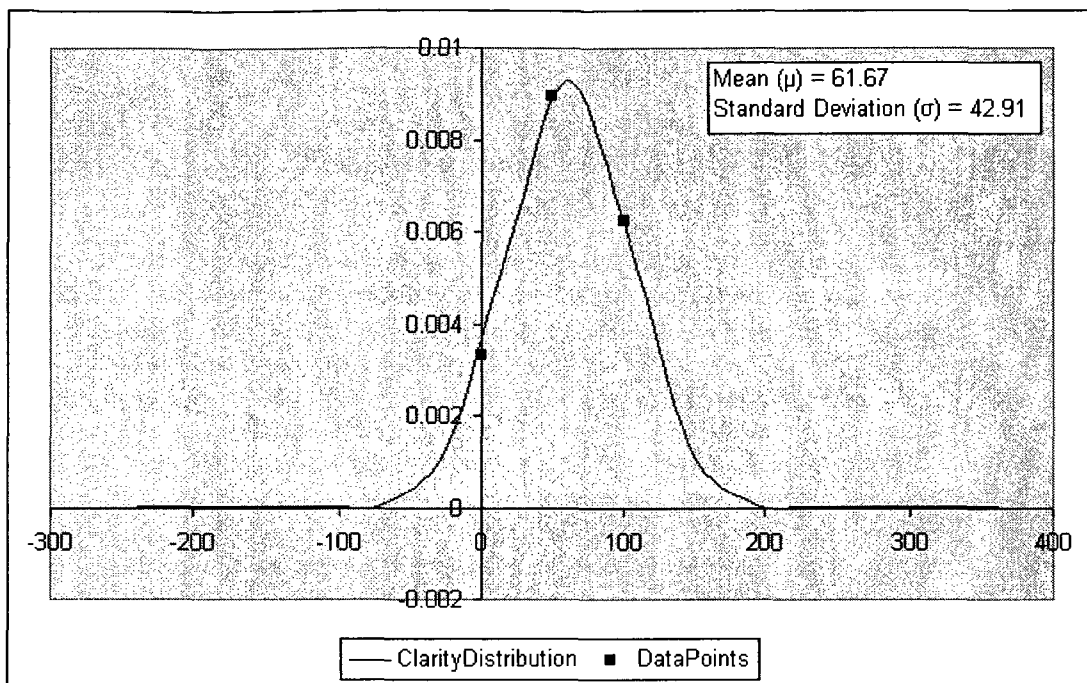


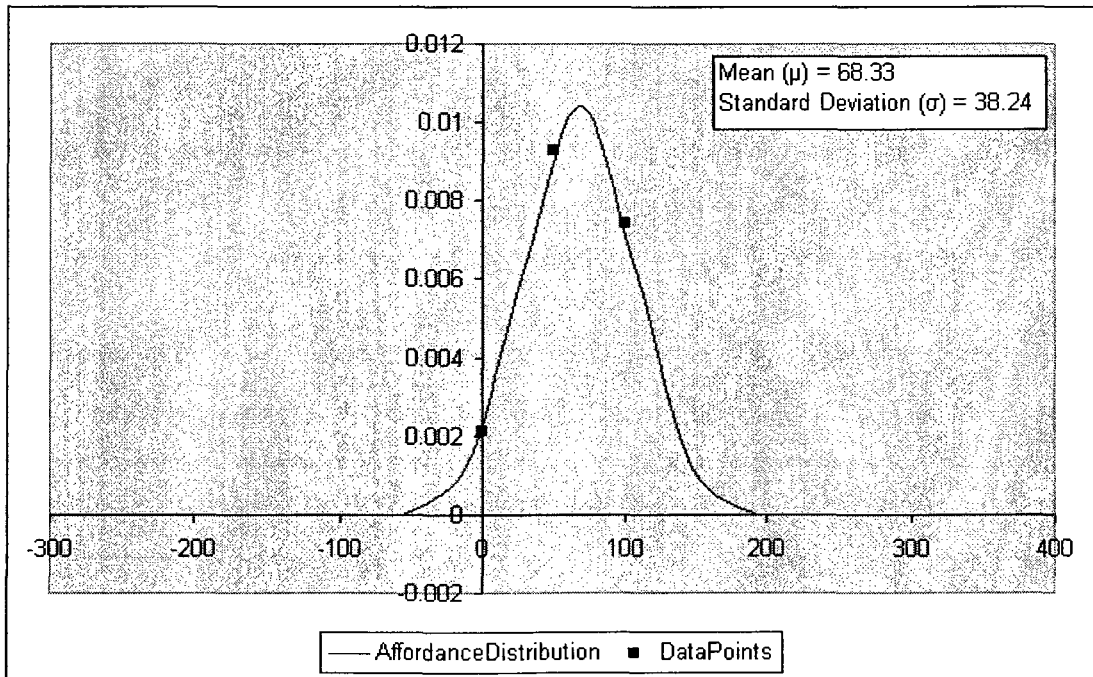
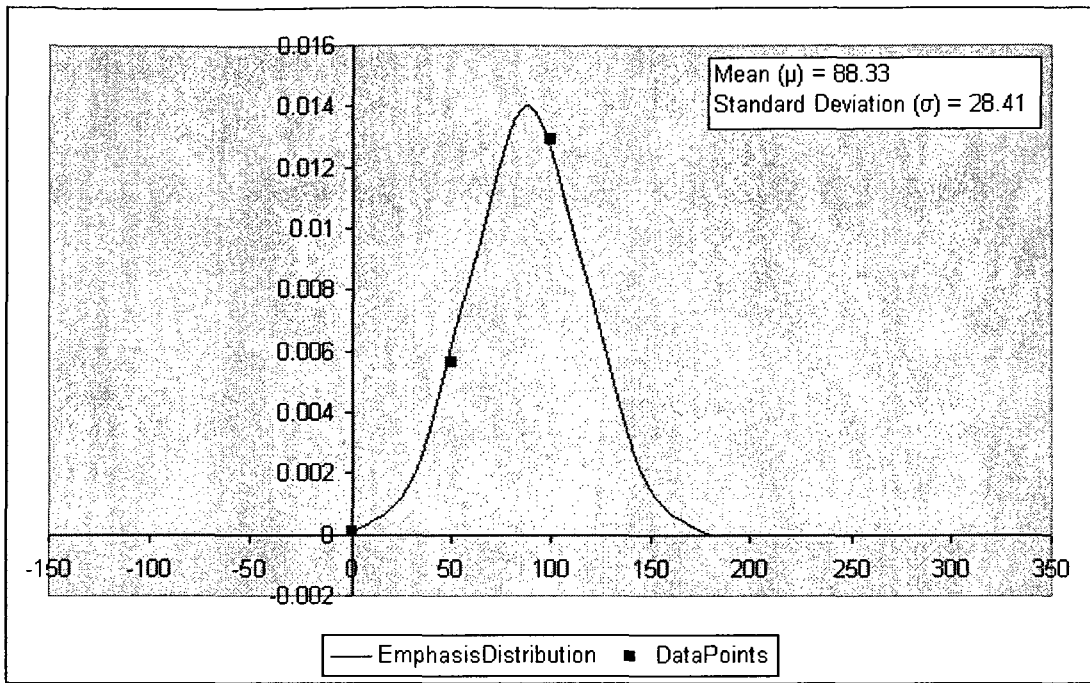


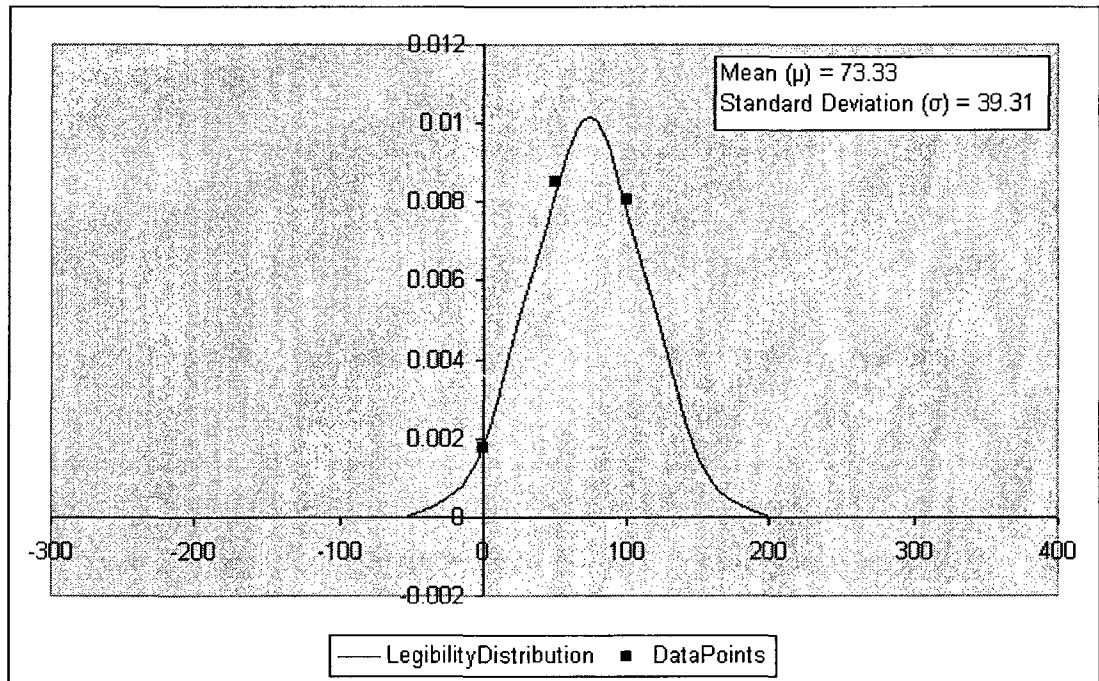
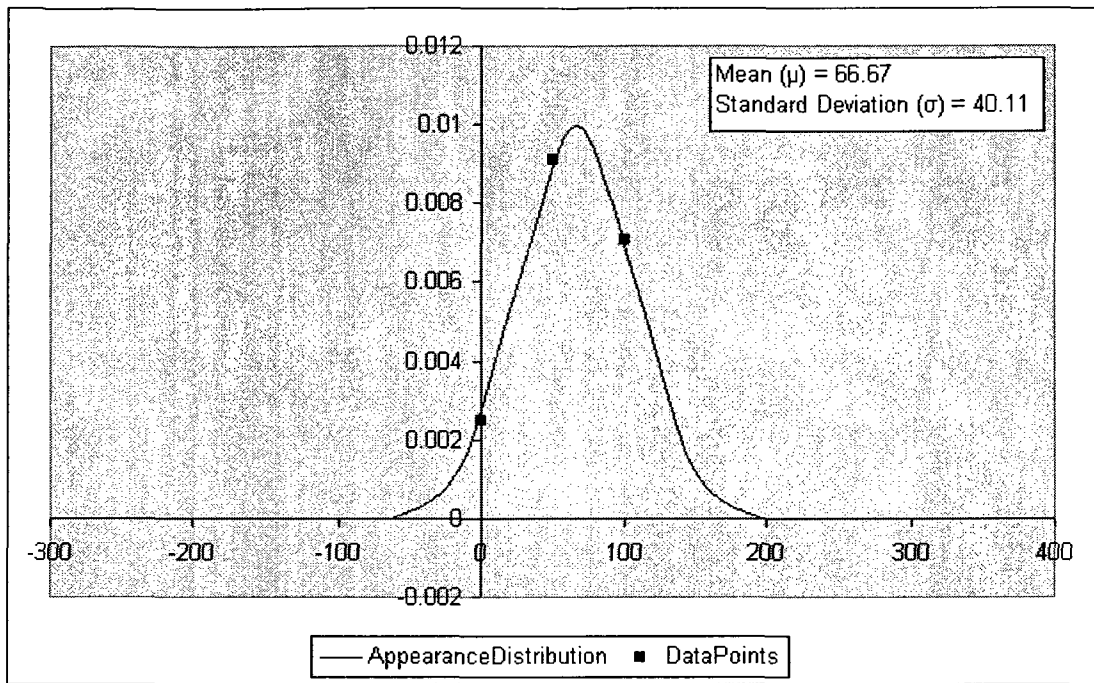


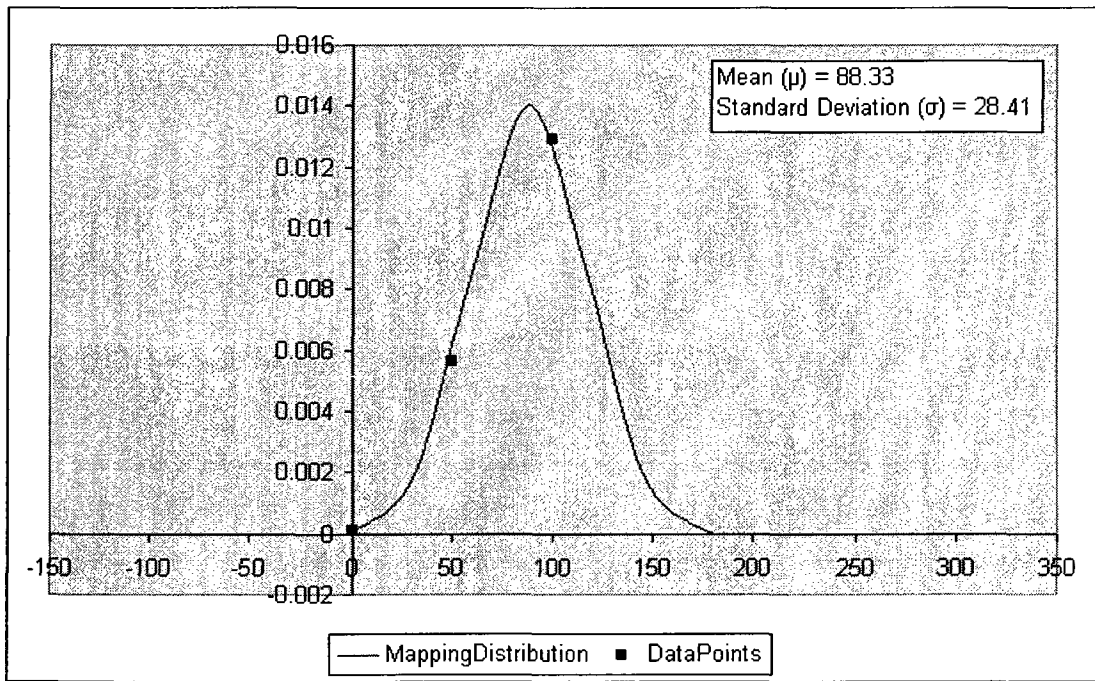
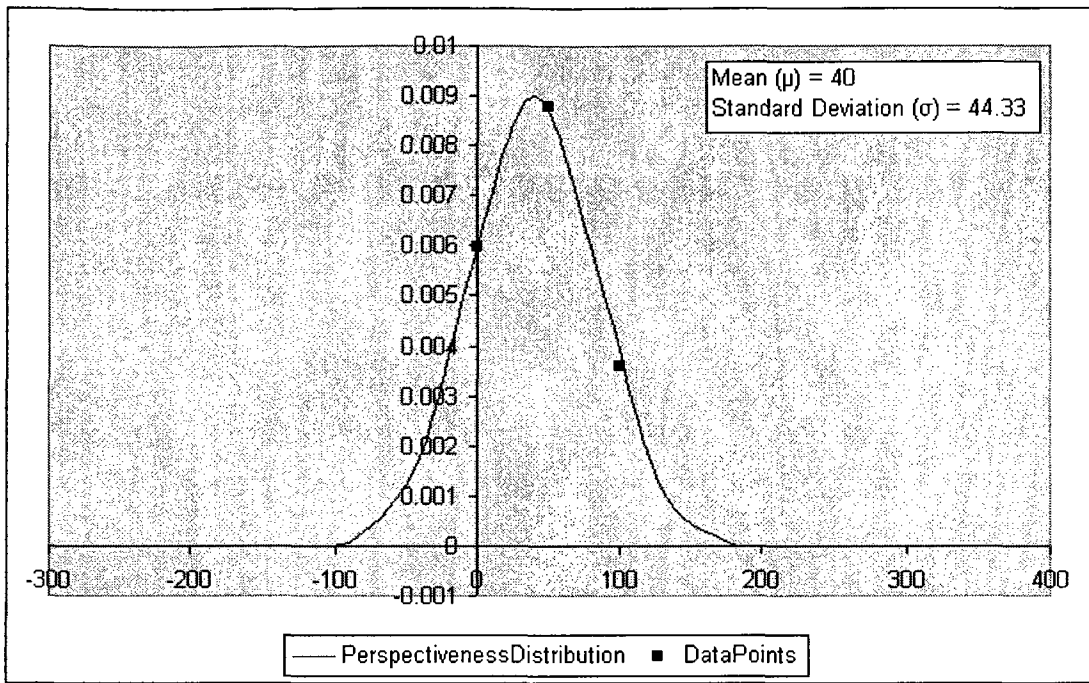
G.4 Normal Distribution Curves for Tree Technique











Appendix H. Analysis of Variance (ANOVA)

Overview

ANOVA is a powerful and versatile statistical technique that is primarily used to compare the means of several groups of observations (Turner and Thayer, 2001). The analysis is based upon the theory that the samples come from normally distributed populations with the same standard deviation. It is assumed that the variable of interest is normally distributed within each group and that each group has the same standard deviation for that variable. The total variance of any sample data set is partitioned into two classes – between-group variability and within-group variability. The between-group variability measures how the sample mean of each group differs from the overall or grand mean. Within-group variability is used to estimate the variation within each group, and it measures the variation about the mean of each group. The main goal in ANOVA is to see whether or not the between-group variability is significantly greater than that of within-group variability. This difference helps to determine if the groups came from different populations or not.

H.1 One-Way ANOVA Test

The total comprehension score for each of the participant is shown in Table H.1. These values are then used to perform one-way ANOVA test to confirm the groupings of participants into experts, intermediates, and novices for each visualization technique as shown in Tables H.2 to H.5. In all the tables from H.2 to H.5, the degree of freedom between groups is ‘ $k-1$ ’ and degree of freedom within-groups is ‘ $N-k$ ’ (where ‘ k ’ is the number of groups i.e. 3 in our case, and ‘ N ’ is the number of participants i.e. 15)

Table H.1: Total Comprehension Score of Each Participant

Participant#	Category M: Male F: Female	Total Comprehension Score			
		Radial Technique	Pyramid Technique	NestedView Technique	Tree Technique
	E: Expert I: Intermediate N: Novice				
1(F)	E	87.5	56.67	58.34	85.84
2(M)	E	83.3	64.16	69.17	74.16
3(M)	I	80	50	70.83	67.5
4(M)	E	82.5	64.16	81.67	77.5
5(M)	I	74.17	62.5	60.83	56.67
6(F)	E	85	60	69.17	72.5
7(F)	E	83.33	49.17	65.84	70.01
8(M)	I	84.16	64.16	80.83	80.83
9(M)	N	95	43.33	75	58.33
10(F)	I	68.33	50	65.84	69.17
11(M)	N	85	81.67	81.67	78.34
12(F)	E	95	60.84	63.33	85.83
13(M)	N	56.67	36.67	53.34	65.01
14(F)	I	69.16	45.83	79.16	60.84
15(F)	N	71.67	55.83	72.49	71.67

Table H.2: ANOVA Results for Radial Technique

Radial Technique Summary						
<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>		
Experts	6	516.63	86.105	22.17055		
Intermediates	5	375.82	75.164	47.01363		
Novices	4	308.34	77.085	276.563		
ANOVA						
<i>Source of Variation</i>	<i>Sum of squares (SS)</i>	<i>Degrees of freedom (df)</i>	<i>Mean square variance (MS)</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	374.5079	2	187.254	1.991011	0.179183	3.885294
Within Groups	1128.596	12	94.04968			
Total	1503.104	14				

Table H.3: ANOVA Results for Pyramid Technique

Pyramid Technique Summary						
<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>		
Experts	6	355	59.16667	31.90559		
Intermediates	5	272.49	54.498	68.24612		
Novices	4	217.5	54.375	394.1977		
ANOVA						
<i>Source of Variation</i>	<i>Sum of squares (SS)</i>	<i>Degrees of freedom (df)</i>	<i>Mean square variance (MS)</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	80.34918	2	40.17459	0.298491	0.747288	3.885294
Within Groups	1615.106	12	134.5921			
Total	1695.455	14				

Table H.4: ANOVA Results for NestedView Technique

Nested View Technique Summary						
<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>		
Experts	6	407.52	67.92	61.87168		
Intermediates	5	357.49	71.498	73.01447		
Novices	4	282.5	70.625	147.794		
ANOVA						
<i>Source of Variation</i>	<i>Sum of squares (SS)</i>	<i>Degrees of freedom (df)</i>	<i>Mean square variance (MS)</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	38.32758	2	19.16379	0.220105	0.805603	3.885294
Within Groups	1044.798	12	87.06653			
Total	1083.126	14				

Table H.5: ANOVA Results for Tree Technique

Tree Technique Summary						
<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>		
Experts	6	465.84	77.64	46.21652		
Intermediates	5	335.01	67.002	85.22057		
Novices	4	273.35	68.3375	74.12596		
ANOVA						
<i>Source of Variation</i>	<i>Sum of squares (SS)</i>	<i>Degrees of freedom (df)</i>	<i>Mean square variance (MS)</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	367.1706	2	183.5853	2.773392	0.102306	3.885294
Within Groups	794.3428	12	66.19523			
Total	1161.513	14				

Using the same tabular data as depicted in Table H.1, the one- way ANOVA test is performed for two groups of 7 females, and 8 males having 1 degree of freedom between groups and 13 degrees of freedom within group as shown in Table H.6.

Table H.6: ANOVA Results for Females' and Males' scores

Visualization technique	F	P-value	F crit
Radial	0.000332	0.985735	4.667193
Pyramid	0.547128	0.47264	4.667193
NestedView	0.730557	0.408179	4.667193
Tree	0.668854	0.428187	4.667193

H.2 One Factor Repeated Measure ANOVA

One factor repeated measure ANOVA was applied to test the variation in the sample of participants for each of the individual technique, as the same participants were repeatedly testing each technique. The tabular data depicted in Table H.1 is rewritten in Table H.7 for the purposes of following calculations.

Number of participants = 15

Number of techniques = 4

Therefore N (total number of scores) = 15 *4 = 60

ΣX or T (i.e. sum of all the scores) = 4167.49

Therefore, $T^2/N = 289466.2$

ΣX^2 (i.e. sum of squares of all scores) = 299249

Table H.7: Comprehension Score of a Participant for Each Technique

Participant#	Radial	Pyramid	NestedView	Tree	RowSUM
1	87.5	56.67	58.34	85.84	288.35
2	83.3	64.16	69.17	74.16	290.79
3	80	50	70.83	67.5	268.33
4	82.5	64.16	81.67	77.5	305.83
5	74.17	62.5	60.83	56.67	254.17
6	85	60	69.17	72.5	286.67
7	83.33	49.17	65.84	70.01	268.35
8	84.16	64.16	80.83	80.83	309.98
9	95	43.33	75	58.33	271.66
10	68.33	50	65.84	69.17	253.34
11	85	81.67	81.67	78.34	326.68
12	95	60.84	63.33	85.83	305
13	56.67	36.67	53.34	65.01	211.69
14	69.16	45.83	79.16	60.84	254.99
15	71.67	55.83	72.49	71.67	271.66
ColumnSUM	1200.79	844.99	1047.51	1074.2	

$SS_{\text{BETWEEN PARTICIPANTS}} = ((\text{sum of squares of all RowSUM scores})/\text{Number of techniques}) - T^2/N = 2902.728$

$SS_{\text{TECHNIQUES}} = ((\text{sum of squares of all ColumnSUM scores})/\text{Number of participants}) - T^2/N = 4339.623$

$SS_{\text{PARTICIPANTS * TECHNIQUES}} = \sum X^2 - T^2/N - SS_{\text{BETWEEN PARTICIPANTS}} - SS_{\text{TECHNIQUES}} = 2540.47$

$df(\text{BETWEEN PARTICIPANTS}) = \text{Number of participants} - 1 = 14$

$df(\text{TECHNIQUES}) = \text{Number of techniques} - 1 = 3$

The ANOVA results for all the four visualization techniques is shown in Table H.8

Table H.8: One Factor Repeated Measures ANOVA Results

<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>		
Radial	15	1200.79	80.05267	107.3646		
Pyramid	15	844.99	56.33267	121.1039		
NestedView	15	1047.51	69.834	77.36614		
Tree	15	1074.2	71.61333	82.96524		
ANOVA						
<i>Source of Variation</i>	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
SS _{BETWEEN} PARTICIPANTS	2902.728	14	207.3377			
SS _{TECHNIQUES}	4339.623	3	1446.541	23.91475	<0.01	4.29
Error (PARTICIPANTS * TECHNIQUES)	2540.47	42	60.48739			
Total	9782.821	59				