Towards a Metrics-Based Framework for Assessing Comprehension of Software Visualization Systems

Harkirat Kaur Padda, Ahmed Seffah, and Sudhir Mudur

Department of Computer Science & Software Engineering, Concordia University, Montreal, Canada
{padda,seffah,mudur}@encs.concordia.ca

Abstract. Despite the burgeoning interest shown in visualizations by diverse disciplines, there yet remains the unresolved question concerning comprehension. Is the concept that is being communicated through the visual easily grasped and clearly interpreted? Given the vast variety of users and their visualization goals, it is difficult for one to decide on the effectiveness of different visualization tools/ techniques in a context independent fashion. To capture the true gains of visualizations, we need a systematic framework that can effectively tell us about actual quantifiable benefits of these visual representations to the intended audience. In this paper, we present our research methodology to establish a metrics-based framework for comprehension measurement in the domain of software visualization systems. We also 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.

Keywords: Software visualizations, comprehension, measurement, metrics, cognition, perception, GUI, patterns etc.

1 Introduction

There are many visualization tools/techniques available today, and many more continue to be introduced as computer-based information processing pervades different domains. Visualization is often seen as a way to help people gain insight about large, related and complex information or artifacts. Despite this apparent proliferation, various researchers have seen many shortcomings in existing visualizations tools which tend to considerably reduce their overall value to users. A detailed study of existing literature shows that possible shortcomings include: 'navigational problems', 'improper context', 'lack of evaluation', 'ineffective imagery', cognitive overload', and interaction difficulty'. With these drawbacks, the utility of the visualization systems is questionable and this is the point where we need to seek measurement. The success of any visualization system relies on its support for providing 'user insights' to understand underlying artifact represented through the visual. If the visualization system does not achieve this objective, it is of little use and suffers from poor quality. Clearly, comprehension is the most important aspect that determines the quality of any visualization system. For visualization systems, we define comprehension as the degree to which information represented through visuals can be grasped and interpreted correctly in a specified context.

M.J. Smith, G. Salvendy (Eds.): Human Interface, Part I, HCII 2007, LNCS 4557, pp. 335–344, 2007.

Existing visualization systems, with sheer volume of information, place high cognitive load on the users. They provide little help to interpret the meanings of different visualizations being displayed. Gleaning from the literature, one can see 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. We need to study and answer research questions like- how well the visualization system's intent is met through visuals and interaction techniques; how well the user's intent is met by the visualization system; whether these representations are really effective 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, what is needed is a framework which enables us to systematically carry out studies for measuring the comprehension aspects of visualization tools/ techniques.

To empirically assess the value of visualizations in a practical sense, we need to study their usefulness in a particular field of their usage; which in our case is software visualizations. In the domain of software visualizations, visualization technologies provide graphical abstractions of the huge source code and assert to ease in perception of this invisible entity by giving it altogether a different aspect than that of a source code. In addition, given the vast variety of users (software engineers in our case) and their distinct visualization goals, it is difficult for one to decide on the effectiveness of different software visualization tools/techniques in a context independent fashion. This notion of 'context of use' has become a pandemic in almost all measures of HCI field; usability itself is not independent of this criteria. We plan to deal logically with this influential factor of any evaluation mechanism by conducting an empirical study with five comparable software visualization tools. Ensuring the effectiveness of a software visualization tool/technique involves understanding of how users use it. This is being simulated in our research by using controlled experiment approach with the help of typical users of these tools. Unlike other researchers who have conducted empirical studies to informally judge the strengths and weaknesses of their tools, our goal is to objectively quantify the overall effectiveness in terms of supported user's comprehension. The software visualization tools (i.e. SA4J [3], SHriMP [10], Structure 101[14], Surveyor [15], and VizzAnalyzer [16]), which we are exploring are the static software visualization tools. These are mainly used in academia and industry during software maintenance purposes to extract the structure of any software system. We investigate the issue of evaluating a visualization tool/ technique by encapsulating it in a pattern format describing the applicable 'context of use' that is appropriate for it. The context of use is defined in detail by studying the actual needs, characteristics of software maintainers and matching their needs with the capabilities of different visualization tools/techniques.

Our proposed metrics-based framework will integrate both quantitative and qualitative measures of users' comprehension which are derived from human's cognitive and perceptual capabilities along with the interface features of any visualization system. Our evaluation criterion is based on three well-defined principles for effective visual communication in HCI – principle of organization, principle of economize and principle of communication proposed by Marcus [5] along with Norman's Cognitive Principles [7]. Like other standard software engineering models (i.e. McCall, Boehm, GQM etc.), the proposed framework incorporates a

hierarchical decomposition of users' comprehension factors, their associated criteria, metrics along with their interpretation, as well as overall evaluation context for comprehension assessment in the form of visualization patterns.

The rest of this paper is organized in various sections as follows:
Section 2 describes the general background and related research in the domain of software visualizations, section 3 along with subsections is a detailed explanation of our research methodology towards the establishment of the metrics-based framework, and finally in section 4, we conclude by enumerating the benefits of our approach and the current stage in the development of this framework.

2 Background and Related Work

The two main facets of software visualizations (SV), static and dynamic, are intended to support many different software development activities; and the developers of these tools claim that their usage improves the productivity of their users especially the software maintainers. Without measurement or evaluation in some form, it is very difficult to realize the true value of such visualization tools to the software community. There is still little progress in the evaluation of software visualizations, as most research effort is being spent on the development of yet more novel visualization techniques, ideas and technological innovations rather than judging the effectiveness/usefulness of the currently available SV tools/techniques. In short, the field of empirical investigation of software visualization tools/ techniques is rather immature and only a few researchers have worked informally to characterize and assess the usefulness of these SV tools/techniques. In the following paragraphs, we briefly summarize various related studies conducted by other researchers in the domain of software visualizations.

Bassil and Keller [1] conducted an online survey of software visualization tools using a questionnaire approach. 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 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.

Knight and Munro [4] briefly discuss 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 structures have a considerable affect on the effectiveness of any visualization.

Marcus et al. [6] 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 which was rendered using 3D metaphor of poly cylinders and containers. A total of 35 participants participated in usability study. The participants were divided into two groups: one group answered the questions using sv3D tool and other group responded 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.

Pacione et al. [8] 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.

Storey et al. have performed a number of experiments with software visualization tools [11, 12, and 13]. In these studies, their primary objectives were to: compare the effectiveness of their tool on five usability dimensions, observe different strategies used by participants while comprehending program under study; how the tools were supporting these set of preferred strategies; devise a framework for describing, comparing and understanding visualization tools that provide awareness of human activities in software development and provide feedback for tool developers and researchers. 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 the tool has been evaluated, deployed).

3 Research Methodology Towards Establishing a Framework

As we have seen in the previous section, many researchers have applied different strategies for empirical investigation of SV tools, but without a unified measurement framework. Our goal is to provide a measurement framework that can objectively quantify the overall effectiveness in terms of supported user's comprehension. To achieve this objective, we are proposing a hierarchical framework to properly investigate the issue of comprehension evaluation as shown in Fig. 1. The three bottom layers are similar to the other well-known software engineering models for measuring the software quality. However, the top three layers are the fundamental layers that are needed before conducting any empirical investigation of the SV tools with their users.

Our research methodology is a step by step refinement of these layers, and is described in the following sections.

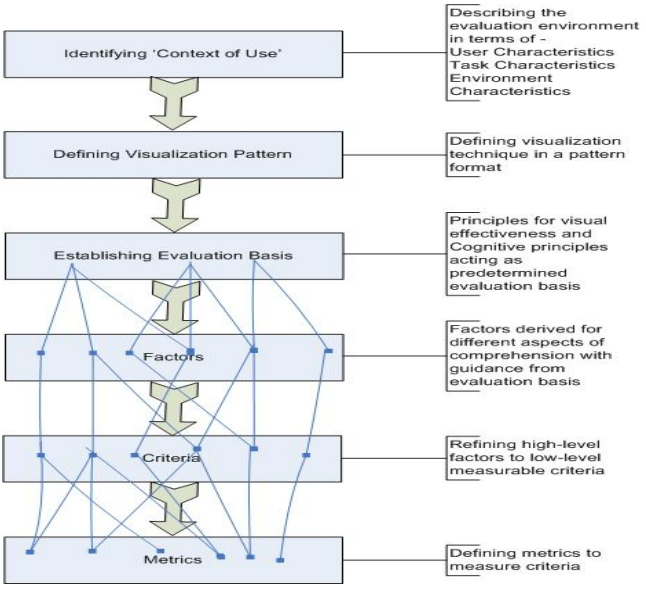


Fig. 1. Research Framework

3.1 Identifying Context of Use

To conduct an evaluation of any SV tool, the foremost step is to identify the 'context of use' that captures the boundaries of evaluation. This is done so that the elements which may influence the evaluation are appropriately summed up. The context of an experiment can be described on three basic dimensions - user characteristics, task characteristics and environment characteristics as briefly described below:

- User characteristics - In our research with static software visualization tools, our users are software engineers. So, the characteristics that have significant impact on their performance are - age, gender, spatial-ability, education, experience (application domain knowledge, programming language expertise, visualization tools expertise) etc.
- Task characteristics - Tasks selected for an evaluation should be representative of what the users (software engineers) do with the static software visualization systems and must be manageable, suitable for a laboratory evaluation. Based on a thorough literature survey, we have identified the needs for software maintenance and have compared the maintenance needs with the tasks supported by current software visualization tools. This is done because the users' tasks that these SV tools support are linked to or are derivable from the typical and elementary information needs of software maintainers. We have developed a catalogue of software maintenance tasks that should be supported by any static software visualization tool. The other characteristics of tasks, like- task type, task size and complexity, task time and cost constraints are also summed up appropriately.
- Environment characteristics - The environment for the experiment is described in detail by determining the appropriate software/ hardware, social components for it.

In our framework, we describe in detail the software part of environment by enumerating different software characteristics like application domain, programming domain, program size, complexity, code quality, availability etc. Hardware characteristics and social characteristics are also studied in detail for experimental purposes.

This 'context of use' is further fed into the next layer of our framework to describe the visualization patterns for visualization techniques used in an experiment.

3.2 Defining Visualization Pattern

The 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 not be appropriate for another purpose [9]. This means we cannot evaluate a visualization technique in isolation without considering the applicable 'context of use'. A technique can be good in one context and bad in another. Wilkins [17] coined the idea of formalizing visualization techniques into patterns by stating that a number of techniques are being reused to solve recurring visualization problems in different domains. That is, we have to evaluate their effectiveness in an abstract manner; by encapsulating the visualization tool/technique in a visualization pattern. We define a visualization pattern as a visualization problem that occurs in a certain context and for which visualization technique can be a solution.

A visualization problem could be solved by a number of different techniques. Consequently, there are many different patterns that could be derived for the same problem. For the tools under our investigation, we are proposing a number of different visualization patterns that have common visualization problem and context as shown in Fig. 2 and Fig. 3. All these visualization patterns are to solve a common visualization

Title	**Radial Tree**
Context	The display consists of a 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	▪ To visualize and navigate large trees in a radial space ▪ Focus + context viewing allowing enormous trees to fit within fixed space of a computer screen ▪ Zoom in on a particular item while keeping in view of the neighborhood context
Solution	Use a Radial Tree representation [3]  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 round the circle and their relationships are shown with directed lines emanating from the source to destination node.
Examples	Sunburst, RadViz
Related pattern	Pyramid/Skeleton View, TreeGraph, Cone Tree, TreeMap, Explorer etc.

Fig. 2. Radial Tree Pattern

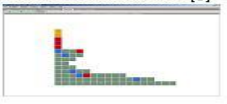
Title	**Pyramid or Skeleton View**
Context	The display consists of a number of software objects (packages, classes and interfaces) and their inter-relationships or structural dependencies in the source code
Problem	How to display large hierarchical tree structures showing dependencies among software objects?
Forces	▪ 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 items. Users can select an item and then highlight the dependent items
Solution	Use a Skeleton View [3]  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. 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. In this view, a stable system should have a normal pyramid shape. An unstable system may look like an upside down pyramid shape. Tangles, dependents of an entity etc. can be specifically colored.
Examples	Icicle plot
Related pattern	Radial Tree, Tree/Graph, Cone Tree, TreeMap, Explorer etc.

Fig. 3. Pyramid/Skeleton View Pattern

problem of displaying large hierarchical tree structures of dependencies among different software objects. The contributing forces may vary in each pattern; however, to distinguish pattern from one another solution is entirely different in each pattern.

3.3 Defining the Evaluation Basis

Our evaluation criterion is based on the work of two eminent researchers (Aaron Marcus and Donald A. Norman) in HCI community who have their expertise in the field of visual communication and human cognition. We believe that the basic principles proposed by them in their respective areas are the fundamental evaluation objectives that contribute to the overall comprehension of any visualization system regardless of its domain. The three principles proposed by Marcus [5] on visual communication i.e. 'Principle of Organization', 'Principle of Economize' and 'Principle of Communication' along with the criteria proposed by Norman cognitive principles [7] on 'Affordances', 'Mapping', 'Constraints' etc. are the building blocks of our evaluation basis. These guiding principles and criteria are studied further in detail in order to determine their affect on human comprehension. The next layer in our hierarchical framework is actually derived from this evaluation basis.

3.4 Determining the Comprehension Factors and Criteria

For any visualization system, data rendered in visual form is perceived or interacted upon by the user of that system as shown in Fig. 4.

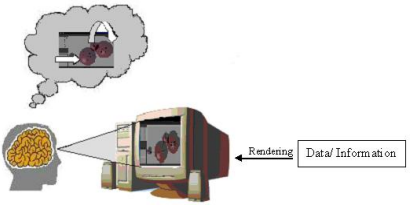


Fig. 4. Aspects of Comprehension

The issue of comprehension evaluation is difficult especially with visualization systems where many different aspects like human cognition, perception, information structure and visualization interface play different roles and affect one another. Brief explanations of each of these aspects are as under:

- Information structure: The information structure has a profound affect on user comprehension. 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. [2]. The net affect is that the visual, which is used to represent it, is not easy to comprehend.
- Visualization interface: Naturally, 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 the correct operations. Based on the Marcus's [5] principle of organization and principle of economize, we have derived a set of factors/ criteria for visualization interface comprehension like consistency, navigability, screen layout, simplicity, clarity/intuitiveness, distinctiveness, emphasis etc.
- Perception: Perception is an integral part of any visualization and perceptual features of visualizations like color, orientation, contrast, position, size etc. along with the criteria proposed by Norman [7] like affordances, metaphors/ symbolism, familiarity can interfere with successful comprehension.
- Cognition: In order to judge the degree of comprehension, it is also necessary to study the human information processing or the cognition of information in human mind. The visualization tool/technique should provide an ergonomic design that matches the cognitive capabilities of the user. To ensure that, Marcus' third principle of communication and Norman's Cognitive Principles guide us to have a number of factors/criteria like legibility, readability, multiple views, naturalness of interaction/mapping etc.

Currently, all the factors in these four aspects of comprehension excluding "information structure" are being studied further in detail to determine their appropriateness to the SV tools. If needed, they can be further decomposed into measurable criteria.

3.5 Determining the Metrics

The lowest level of our framework is a collection of metrics and measures to quantify the related criteria and is currently under implementation. We will be adopting the metrics proposed by various researchers and will define new metrics. We are devising

a set of questions to be asked in a controlled experiment. The questionnaire will incorporate qualitative and quantitative aspects of visualizations which will feed the data for our subjective and objective metrics respectively. Metrics interpretation will be an integrated part of our evaluation framework.

4 Conclusion

In addition to the general measurement benefits, our framework will be a reusable solution that could be applied to other domains in real-world settings to measure comprehension/effectiveness of any visualization system. Specifically, we expect the following benefits:

1. Prior attention to the most important visual design principles for understanding what factors of the visualization system can influence users' comprehension.
2. Provide a flexible hierarchy of the factors and criteria, 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 the 'context of use' and encapsulation template for visualization techniques in terms of the patterns, for better analysis.
4. Data collection efforts will be concentrated, since the required data elements will be already defined.
5. Data interpretation will be more efficient and effectively tied to selected objectives.

Currently, we are refining our repository of factors/criteria and are working on a set of metrics that can be applied for the controlled experiment. We hope that our framework will help and guide other researchers in different domains as well.

References

1. Bassil, S., Keller, R.K.: Software Visualization Tools: Survey and Analysis. In: Proc. of 9th Intl. Workshop on Program Comprehension. Toronto, Canada, pp. 7–17 (2001)
2. Gershon, N.: Visualization of an Imperfect World. IEEE Computer Graphics and Applications 18(4), 43–45 (1998)
3. Iskold, A., Kogan, D., Begic, G.: Structural Analysis for Java (SA4J). An alphaworks Java technology from IBM, (2004), [Accessed October 28, 2006], Available from: < http:// www.alphaworks.ibm.com/tech/sa4j >
4. Knight, C., Munro, M.: Visualisations; Functionality and Interaction. In: Alexandrov, V.N., Dongarra, J., Juliano, B.A., Renner, R.S., Tan, C.J.K. (eds.) Computational Science - ICCS 2001. LNCS, vol. 2074, pp. 470–475. Springer, Heidelberg (2001)
5. Marcus, A.: Principles of Effective Visual Communication for Graphical User Interface Design. In: Human Computer Interaction –Towards the year 2000, 2nd edn., pp. 425–441. Morgan Kaufmann, San Francisco California (1995)
6. Marcus, A., Comorski, D., Sergeyev, A.: Supporting the Evolution of a Software Visualization Tool through Usability Studies. In: Proc. of 13th Intl Workshop on Program Comprehension (2005)
7. Norman, D.A.: The Design of Everyday Things. Doubleday, New York (1990)
8. Pacione, M.J., Roper, M., Wood, M.: A Comparative Evaluation of Dynamic Visualisation Tools. In: Proc. of 10th Working Conf. on Reverse Engg., pp. 1095–1350 (2003)

9. Rushmeier, H., Botts, M., Uselton, S., Walton, J., Watkins, H., Watson, D.: Panel: Metrics and Benchmarks for Visualization. In: Proc. of 6th IEEE Visualization Conference, 422 (1995)
10. SHriMP. The CHISEL Group. University of Victoria, BC, Canada. [Accessed November 2007,2006] Available from: <http://www.thechiselgroup.org/shrimp/manual>
11. Storey, M.A.D., Wong, K., Fong, P., Hooper, D., Hopkins, K., Müller, H.A.: On Designing an Experiment to Evaluate a Reverse Engineering Tool. In: IEEE Proc. of 3rd Working Conf. on Reverse Engg., pp. 31–40, Los Alamitos, CA (1996)
12. Storey, M.A.D., Wong, K., Müller, H.A.: How Do Program Understanding Tools Affect How Programmers Understand Programs? In: Proc. of 4th Working Conf. on Reverse Engg., Amsterdam, Holland, pp. 12–21 (1997)
13. 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: Proc. of ACM symposium on Software visualization, St. Louis, Missouri, pp. 193–202 (2005)
14. Structure101. Headway Software Technologies. [Accessed November 08, 2006], Available from: < http://www.headwaysoft.com/index.php >
15. Surveyor, Lexient Corporation. [Accessed November 2006], Available from: < http://www.lexientcorp.com/codeanalyzer/products.htm >
16. VizzAnalyzer, ARiSA Group, Växjö University, Sweden. [Accessed October 28, 2006], Available from: < http://www.arisa.se/index_projects.html >
17. Wilkins, B.M.: A Pattern Supported Methodology for Visualisation Design. Doctoral dissertation, University of Birmingham, UK (2003)