Published in The 2001 International Conference on Imaging Science, Systems, and Technology (CISST 2001)

June 25-28, 2001, Monte Carlo Resort & Casino, 3770 Las Vegas Blvd., South, Las Vegas, Nevada, USA.

Paper ID: 1038CT
Paper Title: No Silver Bullet for Software Visualisation Evaluation
Paper Author(s): Andrew S. Hatch, Michael P. Smith, Christopher M.B. Taylor and Malcolm Munro

No Silver Bullet for Software Visualisation Evaluation

Andrew S. Hatch, Michael P. Smith, Christopher M.B. Taylor and Malcolm Munro
Visualisation Research Group
Research Institute in Software Evolution
Department of Computer Science
University of Durham
Durham, DH1 3LE, UK

Abstract *Software visualisation seeks to aid comprehension of software; however, there is still little progress in the evaluation of such visualisations. This paper reports on the current practice of evaluation in software visualisation. Four evaluation strategies are identified and discussed in order to identify strengths and weaknesses when applied to software visualisation. Areas for improvement are suggested, particularly with respect to the point at which they are applied, and the need for active interest in evaluation. It is intended that this paper promote discussion and future research whilst demonstrating that, currently, no single method is adequate for a full evaluation.*

Keywords: Software Visualisation, Evaluation

1 Introduction

Software Visualisation (SV) seeks to aid comprehension of a software system by those who are involved with its creation, modification, management and application by using visualisation methods. SV is the application of Information Visualisation (IV) in the Software Engineering (SE) domain in order to offer a means by which software engineers can visually handle the large volume and complexity of information which software systems present. Program comprehension is the traditional motivation for SV, resulting in evaluation strategies which reflect this. This paper aims to highlight the current state of SV evaluation, the relative benefits of different evaluation strategies and the drawbacks that should be considered. As Brooks [1] posed the question about finding a silver bullet for SE, so this study ascertains whether there is a silver bullet for evaluating software visualisations.

2 Background

Evaluation is a key component of any scientific activity and is essential to the development of the SV field. Without measurement, in some form, it is very difficult to realise the benefits of achievements in SV and allow development of a theoretical underpinning to the discipline. For new ideas, evaluation helps determine which are of practical benefit to users, which of those are a good foundation for future research, and those which are of little value. Evaluation is also useful in the comparison of multiple visualisations resulting in an indication as to the most suitable visualisation for the required tasks. Lack of evaluation leads to researchers re-treading old ground as poor ideas are recycled through an absence of filtering.

Presently, evaluations of software visualisations are poor. Most research effort is focused on the development of visualisation ideas, technology and application rather than the development or application of evaluation methods. Research in evaluation is considered to be unattractive resulting in a discrepancy between new developments in visualisation and current evaluation strategies. This leaves few useful frameworks and guidelines although there are a few notable exceptions from Storey *et al.* [2] and Petre *et al.* [3]. Hence, there is currently little support for evaluating new visualisations

successfully, particularly those applied to new fields in SE, or those using new visualisation techniques. Unfortunately, this leads to some visualisations failing to meet older evaluation criteria and being rejected, even though the ideas are sometimes promising. Some evaluation methods developed in the IV community can be applied to SV [4] [5], although they are often too general to be used exclusively.

Apart from the lack of support, evaluation is considered difficult and, depending on strategy, too time consuming to justify the investment. Some evaluation strategies, such as collecting empirical evidence, require resources which are unavailable in terms of participants (subjects and researchers), time and supporting technology. As software visualisation is pushing into new areas so traditional SV evaluation methods are becoming less appropriate. Contributing to all of the above problems is the possibility that researchers are not familiar with areas such as cognition and perception in sufficient detail.

Kitchenham and Jones [6] identify several methods of general SE research evaluation, many of which are used in SV evaluation. However, different methods are suitable to different stages of visualisation development, each with their own advantages and disadvantages. It is important to realise there is no silver bullet for software visualisation evaluation, and there is a need to highlight the issues.

3 Issues in Evaluation Techniques

This section examines the implications of different evaluation techniques, which are considered alongside suggestions on how to improve current practice and highlight areas in which SV evaluation requires development.

A common call within the SE community is the need for empirical evidence to support research findings. Although empirical evidence is good, it is also important to seat this in a strong theoretical background upon which design guidelines and evaluation strategies can build. These guidelines and strategies allow for evaluation of ideas in early stages of visualisation development before empirical studies can be applied. In this way, poor visualisations are rejected before they incur large investment. Current evaluation frameworks, being biased to program comprehension and other lower level SE activities, are becoming less suitable to emerging visualisation arenas such as architecture visualisation.

Deficiencies in evaluation methods lead to the misapplication of design guidelines and taxonomies to the evaluation process, for example the use of Shneiderman's tasks [4] by Wiss *et al.* [7]. Whilst there are a number of taxonomies for software visualisation, such as that by Price *et al.* [8], they are purely intended for classification and not for assessing merit.

Other general issues applicable across all strategies are: the need for consistent terminology for both the strategies themselves (framework versus design guideline) and terms used to describe visualisation components; consideration of the application of strategies to both partial and completed systems in order to filter substandard systems whilst extracting novel concepts; and special consideration of the need for training for particular visualisation systems, especially when these use new and unfamiliar methods and technology.

In order to examine the following four evaluation strategies, a simple model of a SV life-cycle is presented. This is purely to facilitate this comparison. A visualisation starts with the initial idea, perhaps derived from requirements, and progresses to the initial visualisation concept defined in terms of mappings, metaphor, representation and so on. A prototype is then generated for demonstrating main features before the final visualisation is completed.

Each of the four evaluation strategies are presented for discussion of their strengths and weaknesses, application to stages in the SV model above, and suggestions for future direction of the evaluation research.

3.1 Design Guidelines and Frameworks

Beginning with the early stages of the life-cycle of a visualisation, guidelines and frameworks are used extensively within the design process. Design guidelines are general pointers which describe important issues to consider in the creation of visualisation systems, but are not formalised into frameworks. For example, Tufte [5] gives several indicators concerning visual presentation of quantitative data including definitions and principles, whereas Young and Munro [9] are more concerned with areas to be considered when designing 3D software

visualisations. Frameworks, although primarily intended for the analysis of a visualisation system, invariably become design guides.

As the name suggests, these design guidelines promote reasoning about current ideas and visualisation concepts in order to quickly determine if they are valid, useful and incorporate the correct facilities. For example, guidelines can identify key areas in cognitive issues, navigation support, performance requirements and so on. In this way, guidelines help the researcher to filter out the development of bad ideas, saving resources.

There are problems with the adoption of design guidelines and using frameworks for design purposes when those that are used are outdated and newer alternatives are unavailable. For example, Shneiderman's seven tasks of information visualisation [4] describe the steps of overview, zoom, filter, details-on-demand, relate, history and extract. For non-static data, especially rapidly changing data, these guidelines become difficult to evaluate against. Considering the zoom step, how would this be affected if data is constantly changing and data is added which should be part of the display?

Some researchers have taken design guidelines and applied them as though they were an evaluation framework. It is not the intention of some guidelines to be used as an evaluation framework as they are usually general and open to a large range of interpretation. For example, some authors [10] collect together frameworks and guidelines and perform an evaluation based on this. By designing a visualisation system with a set of criteria, and then evaluating that system using the same criteria, there is a possible danger of self-measurement in that the system is designed for the evaluation, not the original requirements. Guidelines are often open-ended and are designed to be applicable in a broad sense, therefore evaluating against these guidelines can lead to justification of a substandard property. For example, Shneiderman's *relate* property – which states that the visualisation should be able to "view relationships among items" [4] – can be taken in a very broad sense and applied to any number of visualisation functions. Further, by taking guidelines out of context and applying them in areas to which they are unsuited an evaluation can be invalid. Whilst there is not a problem in demonstrating that a visualisation adheres to good guidelines, it should not be presented as an evaluation, especially in the cases where the guidelines are generic and open to differing interpretations.

Application of guidelines as evaluations can be attributed to a lack of good evaluation frameworks in the field, and this will be considered in the next section. It is important that the community develop new guidelines in order to deal with the new branches of software visualisation research such as real-time, dynamic, system and architecture visualisation. There are useful generic guidelines in IV as a whole, but it is desirable to have specific task-related guidelines for software visualisation.

3.2 Feature-Based Evaluation Framework

A feature-based evaluation framework is a popular evaluation strategy due to the ease in which it may be applied. As mentioned by Kitchenham and Jones [11], such a framework imposes no prerequisites on infrastructure or target system, whilst providing simple screening of systems at multiple levels of detail. A framework is particularly valuable when assessing many ideas, as often they are based on multiple-choice questions that may be answered rapidly. This allows relative benefits of the ideas to be determined, so allowing comparisons to be made, with a low investment. There are many examples of such frameworks, such as Young and Munro [9], and Knight [12].

Despite the apparent simplicity of a feature-based framework, there are many inherent problems that must be resolved in order to implement a successful evaluation. Most obvious is the style of question that should be presented by the framework. A question requesting a simple yes/no answer can be too open ended. For example, Storey *et al.* [2] suggests that a visualisation should "indicate maintainer's current focus". It is easy to argue that the current focus could always be the object at the centre of the screen, despite this not being the original intention of the question. However, a question to be answered on a sliding scale may then become too subjective, with the answer often depending on the experience that the user had with similar visualisations. For example, the answer to a question asking whether a three-dimensional visualisation was easy to navigate would depend greatly on the previous experience that the user had working within such environments. Further issues arise when the question is simply too vague to answered

accurately. An illustration of this is where Wiss and Carr [13] asked users whether a visualisation was simply 'good' and 'easy to use' with responses on a sliding scale.

Current frameworks also fail to consider 'negative features' - unwanted features within the visualisation that have a detrimental effect overall, such as those considered by Globus and Uselton [14]. Gestalt effects are significant, and can distort the mental view that the user has of the visualised data. Similar distortions may arise through the use of animation, colour, size and many other features. Failure to consider such features when evaluating using a framework can lead some visualisations to be considered acceptable, when they are fundamentally flawed in practice. Other important features currently not covered by most frameworks include issues of scalability and accuracy. Many current frameworks do not consider aspects highlighted by Tufte [5], such as complexity, density and beauty. If these properties are important to the visualisation then other methods of evaluation such as guidelines or user study should be used.

A feature-based evaluation framework should be applied at all points of the SV life-cycle. However, unless the framework includes contradicting ideals, such as low complexity with high information content, care should be taken to avoid detailed evaluation at the conceptual stage, as otherwise the visualisation may become tailored to satisfy the evaluation criteria.

Despite the number of frameworks that exist, many of these are based on the framework by Storey *et al.* [2] or Shneiderman's guidelines [4]. In order to counter the problem of modifying frameworks to suit a visualisation, which may become a self-measuring process, it is clear that more frameworks are required. Although new generic frameworks would be beneficial in order to consider some of the new trends within IV generally, it is also necessary to create more domain-specific frameworks to reduce the problems of tailoring frameworks whilst maintaining suitability and relevance.

3.3 Scenarios and Walkthroughs

Scenarios and walkthroughs, whilst not a traditional evaluation method, offer a showcase for demonstrating features of a visualisation, as used by Chi *et al.* [15] for example. They move the burden of the evaluation onto the reader, allowing them to evaluate the visualisation in terms of their own experiences and requirements. This is often done alongside other evaluation strategies in order to demonstrate usage rather than to specifically assess the visualisation. For example, work by Knight [12] shows how information on determining the impact of a change to the type of a class variable can be found in the Software World visualisation, alongside a feature-based evaluation framework.

The use of scenarios and walkthroughs presents a number of advantages and disadvantages. They offer a more concrete example of the usage of the visualisation for the presented task and in a more natural way than can be obtained simply from a description of the visualisation in terms of mappings, metaphors and representation. They often include many screenshots as evidence, allowing an impression of the user-interface, technology and visual quality to be gained. This allows the reader to make a more informative judgement on the merit of the visualisation for the given task, by placing the emphasis for evaluation on them and therefore reducing the bias of self evaluation. However, this means that evaluation is individual to the reader and subject to their own bias. This individual evaluation is not propagated to the wider SV community in terms of a concrete evaluation that can act as a building block for future work. A storyboard of images can provide much information about the visualisation, nevertheless it is still hard to get a feel for some issues such as navigation, especially for example, in 3D visualisations. Scenarios and walkthroughs can illustrate aspects of the visualisation in more detail than frameworks, however, the features demonstrated will often only show the visualisation in a positive light, highlighting well supported tasks with favourable data and conditions. Only a limited number of tasks can be covered due to the verbose nature of detailed explanation and images, even in large publications such as theses. The problem becomes worse for smaller scale publications. The subset of tasks covered allows poorly or unsupported tasks to be hidden.

Scenarios and walkthroughs should not be the primary focus of an evaluation, due to the large possibility of bias by the researcher, and the limited subset of the visualisation demonstrated. However, they are relatively easy to produce and can offer a way to check the visualisation, from the initial idea to the final

visualisation, against support for some required tasks.

Future effort could be invested in increasing the use of specific tasks by basing task selection on research into information requirements for specific activities, such as those on program comprehension by Mayrhauser and Vans [16]. These requirements can then be verified more appropriately than by using a framework, as scenarios and walkthroughs show the process by which the information can be found, which can be just as important as showing that it is actually present.

3.4 User and Empirical Studies

Empirical studies attempt to provide hard evidence to support hypotheses, for which there has been an increasing call within the SE community. Some researchers have applied empirical studies to IV and SV, especially when measuring the time taken to complete tasks. There are, however, comparatively few good attempts to use this strategy for evaluation in the SV field. One strength of empirical studies is their ability to allow statistical analysis of results. This enables comparison of the unsupported task against the use of a visualisation to support it, along with cross visualisation comparisons. A common test is to measure the time taken to do a particular task with and without visualisation support in order to give quantifiable results such as 'the use of the visualisation enables the task to be completed three times quicker'. Statements like these are more supportable than statements which describe properties of a visualisation, such as 'it allows the user to maintain focus and context'.

User studies can be carried out independently or as part of an empirical study. By using questionnaires and user observations, user studies record individual and collective user-experience, as applied by Storey *et al.* [17] for example. These kinds of studies are useful in contrast to empirical studies with users as they highlight individual issues that might be overlooked when combining results of a number of users. Nevertheless care must be taken in generalising individual user-experience.

User and empirical studies have the benefit of helping to reduce the bias of self evaluation, but this introduces the overhead of having to train users in the use of the visualisation. Failure to train sufficiently can yield results which are influenced by this confounding factor. Many other influences can also confound results, most notably user experience. This can have a significant effect on experimental results due to user differences in knowledge in the domain, and familiarity with the task and environment. Users bring their own bias, especially to visualisations based on techniques such as 3D interfaces where ability to learn, openness to new techniques and spatial awareness can make an impact on the results. Common sources of error lie with the type and scale of tasks being set, and also subject selection problems, for example students are often used rather than experienced programmers for the SV domain [17] . One mistake is to then take these results and generalise them beyond experimental conditions, for example, applying student-based studies to professional programmers and academic environment-based studies to industrial organisations. In order to counteract the problems highlighted here, considerable investment in time and organisation is required. This can make such studies unattractive for some situations, particularly small or short-term projects.

Application of these studies relies on the existence of a partial or full implementation of a visualisation system and the infrastructure and resources to support such experimentation. For industry to adopt visualisation strategies and tools, empirical evidence is preferred, although this is not always required. To this end, larger-scale problems which span longer time-frames should be considered in evaluations.

4 Conclusions and future directions

The goal of the work presented here is to address the current position of evaluation within the SV field, and to suggest future research. By analysing the current evaluation process three related areas of concern have arisen. Firstly, the majority of SV research effort is concentrated on the development of visualisation ideas, technology and application rather than the development or application of evaluation. Secondly, this has led to a deficiency in the quality and quantity of available evaluations. Finally, this has an impact on the misapplication of existing evaluations methods in an attempt to perform an evaluation.

Four evaluation strategies have been examined in terms of relative benefit, current

usage and scope for future development. The primary conclusion is that more work is required in all areas. In particular, it has been identified that these strategies can be selectively applied to different stages of a software visualisation in order to cultivate good ideas efficiently, whilst filtering poor ones. Each of these areas requires effort in order to ensure that the strategies can be applied to visualisations based around new methods and technologies.

The principal problem in the use of design guidelines is their misapplication as an evaluation. Whilst guidelines are useful for checking the presence of high level ideals, it is too easy to justify conformance to these to be a sole means of evaluation. A feature-based framework should offer more specific assessment. However, the current shortage of suitable frameworks means that many are based around design guidelines only, or tailored to specific visualisations creating an atmosphere open to self-measuring evaluations. More relevant frameworks are required in order to counter this problem. Scenarios and walkthroughs offer an indirect evaluation of a visualisation for specific tasks, by leaving the decision of merit up to the reader. Hence they should not be used as the main form of evaluation, but could be beneficial for SV evaluation especially if combined with increased research into information requirements within the SE domain. Empirical and user studies offer the ability to distance the researcher from the evaluation although the possibility for bias is still present. As noted, industry is more likely to adopt visualisation if sound empirical evidence is demonstrated. In order to provide this, larger-scale problems which span longer time frames may become necessary.

In general the software visualisation field should be more prepared to adopt suitable evaluations developed by applicable fields, such as software engineering, information visualisation and human computer interaction.

It is clear that this study has raised more questions than it has answered. This was the intention in order to promote discussion within the SV community. What is apparent however, is that there is currently no silver bullet for evaluating software visualisations.

Acknowledgements

Thanks to Claire Knight for her helpful comments on this paper. Andrew Hatch and Michael Smith are supported by EPSRC research studentships.

References

[1] F.P. Brooks, "No silver bullet: essence and accidents of software engineering", *IEEE Computer* vol 20, 1987 pp.10-19

[2] M.-A.D. Storey, F.D. Fracchia, and H.A. Müller, "Cognitive Design Elements to Support the Construction of a Mental Model during Software Exploration", *Journal of Software Systems*, Vol 44, 1999. pp. 171-185.

[3] M. Petre, A.F. Blackwell, and T.R.G. Green, "Cognitive Questions in Software Visualisation" *Software Visualisation: Programming as a Multi-Media Experience*, J. Stasko, J. Domingue, M. Brown & B. Price eds. MIT Press, January 1998. pp. 453-480

[4] B. Shneiderman, *Designing the User Interface Third Edition: Strategies for Effective Human Computer Interaction*, Addison-Wesley, 1998, pp. 522-541

[5] E. R. Tufte, *The Visual Display of Quantitative Information*, Graphics Press, February 1992 reprint.

[6] B. Kitchenham and L. Jones, "Evaluating Software Engineering Methods and Tools. Part 1: The Evaluation Context and Evaluation Methods", *Software Engineering Notes*, Vol. 21, No. 1, Jan. 1996. pp. 12-15.

[7] U. Wiss, D. Carr, and H. Jonsson, "Evaluating 3-Dimensional Information Visualization Designs: a Case Study", *Proc. IEEE Conference on Information Visualization*, London, England, July 29-31, 1998, pp. 137-144.

[8] B.A. Price, R.M. Baecker, and I.S. Small. (1993). "A principled taxonomy of software visualization", *Journal of Visual Languages and Computing*, Vol. 4, No. 3, 1993. pp. 211-266.

[9] P. Young and M. Munro, "Visualising Software in Virtual Reality", Proc. 6th International Workshop on Program Comprehension (IWPC '98), Ischia, Italy, June 1998. pp. 19-26

[10] J. Huotari, "Supporting user's understanding of complex information spaces by advanced visualisation techniques". *STeP'98 - Human and Artificial Information Processing*, Jyväskylä, Finland, 1998. pp. 41-50

[11] B. Kitchenham and L. Jones, "Evaluating Software Engineering Methods and Tools. Part 5: The Influence of Human Factors", *Software Engineering Notes*, Vol. 22, No. 1, Jan. 1997. pp. 13-15.

[12] C. Knight, "Virtual Software in Reality", PhD Thesis, Dept. of Computer Science, University of Durham, 2000

[13] U. Wiss, and D. Carr, "An Empirical Study of Task Support in 3D Information Visualizations", *Proc. IEEE Conference on Information Visualization*, London, England, July 14-16, 1999, pp. 392-399.

[14] A. Globus, and S. Uselton, *Evaluation of Visualization Software*, Report NAS-95-005, Computer Science Corporation, NASA Ames Research Center, 1995

[15] E.H. Chi, J. Pitkow, J. Mackinlay, P. Pirolli, R. Gossweiler, and S.K. Card. "Visualizing the Evolution of Web Ecologies". *Proc. ACM CHI 98 Conference on Human Factors in Computing Systems*, ACM Press, Los Angeles, California, 1998. pp. 400-407

[16] A. von Mayrhauser, and A.M. Vans, "Program Understanding Behaviour During Adaption of Large Scale Software" *6th International Workshop on Program Comprehension*, IEEE Computer Society, Ischia, Italy, 1998, pp.164-172

[17] M.-A.D. Storey, K.Wong, and H.A. Müller, "How Do Program Understanding Tools Affect How Programmers Understand Programs?", *Science of Computer Programming* vol 36, March 2000. pp. 183-207