

METHODOLOGIES FOR STUDIES OF PROGRAM VISUALIZATION

Niko Myller & Roman Bednarik
Department of Computer Science
University of Joensuu
PO Box 111, FI-80101
firstname.surname@cs.joensuu.fi

ABSTRACT

Learning and interaction with program visualization tools is a complex domain that needs to be approached from various perspectives. While most of the studies conducted in past made use of controlled experiments, other types of approaches such as classrooms studies were missing. The main claim of this paper is that in order to holistically investigate this complex domain, multiple perspectives and methodologies have to be involved. A number of qualitative and quantitative studies have been conducted by us and others to research a program visualization tool called Jeliot 3. We summarize some of the results and discuss methodological lessons learned when conducting these types of studies.

INTRODUCTION

In their review of the algorithm visualization research, Stasko and Hundhausen (2005) argue for using evaluation methods other than controlled experiments. Considering the number of studies of algorithm visualization that made use of controlled experiments, this call is justified. However, we believe that in order to holistically evaluate the outcomes of visualization, researchers have to triangulate the data collected from multiple sources with several methodologies.

Stasko and Hundhausen (2005) list and juxtapose five types of empirical studies previously reported: *controlled experiments, observational studies, questionnaires and surveys, ethnographic field techniques, and usability studies*. The authors concentrate on effectiveness of (algorithm) visualizations i.e. how well one performs with the visualization tool compared to others who do not use the tool; we see the domain of visualization studies as extending beyond the effectiveness. Therefore, we define the context of our discussion as studies of human participants interacting and learning with (program) visualization tools.

In this paper, we report on our experience with combining a number of methods to study several aspects of a visualization tool. The rest of this paper is organized as follows: first, we introduce Jeliot 3, a case program visualization environment. We discuss classroom-based approaches and controlled laboratory studies. Questionnaire based data collection and analysis is described and finally the discussion and conclusions are presented.

JELIOT 3

The Jeliot family is a collection of program visualization environments that have been developed over the past ten years (Ben-Ari et al., 2002). The latest version Jeliot 3 (Figure 1) was developed at University of Joensuu (Moreno et al., 2004) and designed to ease teaching programming concepts to novices and aid in program comprehension and debugging. Its predecessor, Jeliot 2000, has been successfully used in introductory programming courses, helping novices to acquire better vocabulary to explain programming concepts and explain unseen situations (Ben-Bassat Levy et al., 2003). This might be due to the fact that the visual and concrete model of Jeliot helps the learner to build a viable mental model of the program execution and use it to verbalize the execution (Mayer, 1981).

While Jeliot 3 is technically one of the most advanced tools to visualize and animate Java programs' execution for novice programmers, its educational and pedagogical benefits have not been fully evaluated, researched and demonstrated. In order to begin to fill this gap, we have conducted a series of experiments

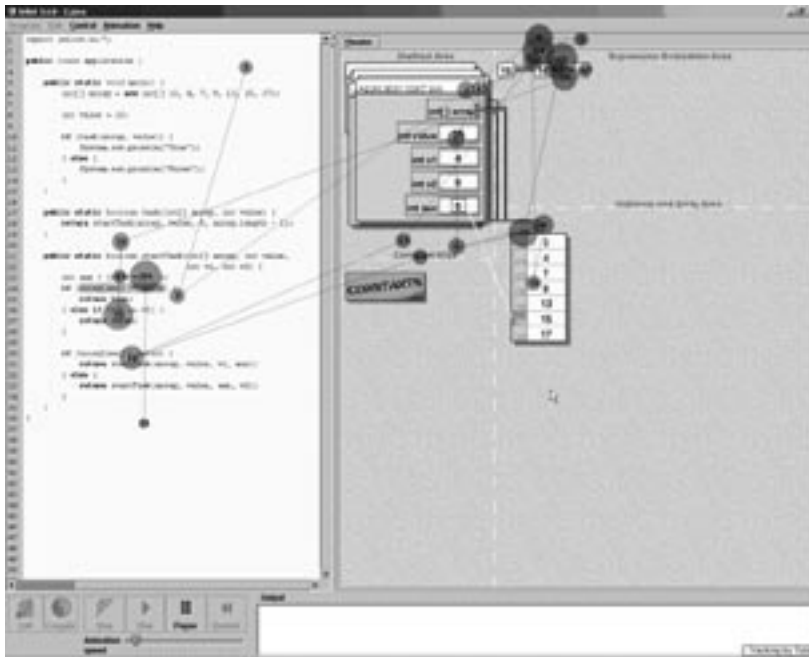


Figure 1. Jeliot 3 screenshot with a typical scan-path superimposed.

that have employed several experimental methodologies. In this paper, we give an overview of and compare the methodological approaches that have been applied to study user interaction and learning with Jeliot 3.

CLASSROOM STUDIES

A number of classroom studies have been conducted in order to evaluate and analyze the usage of tools in Jeliot family by using both qualitative and quantitative methods. Even though classroom studies are more difficult to control, they can give more externally valid results. Thus, they can be used to describe the practices that take place in classrooms and analyze how the tool usage affects them.

A good example of qualitative work done with Jeliot and visualizations is the research by Lattu et al. (2000, 2003). Studies were conducted in several programming courses and data were collected by observations, video recording, and interviews as well as by using questionnaires. The results were presented as categories of visualization utilizations, problems encountered when using the visualizations, types of visualizations used etc. The content of these categories have guided the further development of Jeliot by giving a better idea in what kinds of situations Jeliot can be used and how it is expected to work.

Ben-Bassat Levy et al. (2003) used mixed methods to analyze the impact of using Jeliot 2000 in an introductory programming course at high school. The quantitative findings indicated that Jeliot helped especially the mediocre students to learn programming. More interestingly, the qualitative findings explained this difference because Jeliot had helped the mediocre students to acquire vocabulary to describe programs. This in turn helped students to overcome and explain new situations.

Recently, the first author has conducted a classroom study regarding the collaborative use of program visualization. In this study, the impact of program visualization to collaboration practices will be evaluated. The aim of the study is to identify needs that collaborative use of program visualization tools raise. The materials were collected by video recording students working during laboratory sessions and interviewing them after the course. Also in this context, it seems that just seeing the visualization is not enough, but the visualization

should support meaningful conversations and collaboration between students by providing interaction opportunities during the visualization (e.g. in the form of value inputting or prediction-type questions etc.). This is an extension of the work related to how the engagement of the visualization affects learning in single learner case to the context of collaborative use of visualizations.

Currently, another classroom study has been conducted using close observation of students' needs related to the tool usage in a programming course. The study included notes taking, video and audio recording and interviewing. The aim of the study is to survey what kinds of problems and needs students have during an introductory programming course when they are using a visualization tool. This is done in order to allow adaptation of the tool to different needs of students.

Lessons Learned from Classroom Studies

Higher external validity is achieved on the expense of the control and/or internal validity. This can cause problems especially when quantitative methods are used. For example, a direct comparison of the results obtained in classroom studies to those obtained using quantitative methods is hard. Less control also means that classroom studies need to be designed with more care. In addition, a classroom study needs to be run long enough in order to allow all the effects of visualization to appear. The use of the visualization tool should be extensive in a sense that the teacher should be using it during teaching and students should be using it to complete their exercises both in laboratory sessions and at home. If the tool in question is not actively used in the course and presented by the teacher, but rather employed as an additional means of learning, other confounding factors might prevail and threaten the study.

Another issue related to the programming courses is that currently the drop-out rates are high. This might cause bias to the results when several groups are compared and thus group sizes should be kept higher than otherwise needed. If the unit of analysis is a small group rather than an individual, the members in the groups might need to be changed during the course because of the drop-outs. This in turn might cause problems when the groups are compared longitudinally.

Laboratory Studies

Controlled studies provide researchers with quantitative data that describe certain aspects of investigated tasks. In the context of program visualization, the typical questions one might ask are, e.g., *“What is the difference between novice and expert programmer in how they interact with the visualization tool?”*, or *“Is better comprehension achieved when blocks of source code are highlighted during visualization?”* To answer these and similar questions, the relationship between independent variables (those manipulated by researcher) and dependent variables (measures captured during the task) is investigated using various types of statistical methods.

In a typical laboratory study, a number of subjects that represent the population of interest are recruited and pre-tested. In our experience, technical problems such as data storage failure might occur or based on the pre-test data, some participants might not represent the target population. Therefore, the number of recruited participants shall be kept high enough in case that any unwanted issues appear.

Protocols recorded during controlled experiments include audio and video signals, interaction logs, and performance measures. Audio protocols serve to capture verbalized thoughts during task completion (so called concurrent verbal data) or after the task (retrospective verbal data). For an in-depth treatment of the topic see Chi (1997).

Recently, some studies in the domain of psychology of programming (e.g., Nevalainen and Sajaniemi, 2005; Bednarik, Myller, Tukiainen and Sutinen, 2006) made use of eye-movement tracking techniques to capture the location of visual attention focus of experimental participants. At their present technological development, eye-movement tracking techniques are able to capture the patterns of visual attention of one subject. A typical stream of eye-tracking data contains either a monocular or a binocular timestamped vector consisting of location of participant's attention, pupil size, validation codes and other items such as events that occurred in tracked window of interest.

Two laboratory controlled experiments have been conducted with Jeliot 3, a comprehension study



and a debugging study. Both of them made an extensive use of recording various interaction protocols, including audio and video recording and eye-movement data.

In a study using Jeliot 3 to aid comprehension, (Bednarik, Myller, Tukiainen and Sutinen, 2006) investigated differences between novice and intermediate programmers in terms of behavior and interaction patterns with Jeliot 3. Eye-movement data were recorded in order to examine visual strategies during program animation. It has been found that there are only subtle differences between subjects of various experience levels during the program animation; however, the differences were found in overall interaction with the tool.

Another study that has been conducted in the controlled environment concentrated on debugging strategies of programmers using Jeliot 3 environment. Although Jeliot 3 is not primarily a debugging tool, short programs such as those used in this study can be carefully investigated and trouble-shot. In the study, an unknown program was given to participants together with a textual description of the problem to be solved. Similarly as in the previous controlled studies, audio protocols, interaction data and eye-movement data were collected.

These studies directly informed the design of the visualization and the interface of the tool. For instance, it has been found that those subjects who spent more time on looking at certain areas of the visualization provide corresponding information types in their comprehension summaries (Bednarik & Myller, 2006). Currently, we are analyzing a possible interaction between levels of experience and quality of verbal protocols during the comprehension and debugging tasks.

Lessons Learned from Laboratory Studies

Controlled laboratory experiments are an objective method that is dependent on the statistical analysis method used. A typical problem faced by a researcher using this experimental methodology is that of a sample size. Depending on the number of hypotheses and factors, we recommend that at least twenty participants shall be recruited for one factor study. Some phenomena might not be described using measurable variables, so that a researcher cannot operationalize the outcomes. In those situations, the qualitative, classroom studies are a recommended solution.

Because the domain of computer programming is relatively complex in cognitive terms, laboratory studies that investigate learning and interaction cannot directly apply the settings as seen in the studies of human performance. Imposing strict controls on the actual learning task can cause changes to the behavior of the participants. By loosening the controls of the laboratory studies, for instance by allowing the participants to freely interact with the tool, external validity can be improved. However, increased freedom of interaction might pose new problems to quantitative analysis methods, such as ANOVA because some data might be missing or a slightly different data was obtained from each subject.

It is relatively easy to use various data capturing methods in the laboratory environment compared to classrooms or other more natural environments. For instance, while video protocols can be recorded also in a classroom environment, eye-movement data or verbal protocols are difficult to obtain. Laboratory studies can provide high internal validity, while the external validity is at least partially compromised.

QUESTIONNAIRE STUDIES

Questionnaire-based studies are often conducted to provide researchers with complimentary data. One example of a formal method to collect such data is TUP-based evaluation (Bednarik et al., 2004). The TUP-model stands for the words of Technology, Usability, and Pedagogy. The model is an evaluation scheme which focuses on three aspects of educational environments: The technological aspects concentrate on issues such as the dependencies and interactions between the environment and surrounding software and hardware equipment, issues concerning security and privacy, material sharing and reuse. The usability aspects in educational environments concern into some extent the traditional usability issues or motor and perceptual factors related to interaction with an environment. Finally, the pedagogical aspects of the TUP model refer to the need for evaluating the pedagogical qualities of educational environments.

A TUP-based study is conducted by distributing a questionnaire to a larger number of participants that have been involved in interacting with the environment in study. Automatic collection of the data is provided by an online service. Example evaluations of Jeliot 3, conducted by approximately 18 participants ranging from students to teachers and developers, are accessible online, c. f. <http://cs.joensuu.fi/~tup>.

Questionnaires were also used by Kannusmäki et al. (2004) to collect students' reflections on programming and the utilization of the programming tools during a second course of programming in a web-based study program, ViSCoS. Students filled in every week a web-questionnaire where they answered open-ended questions related to their learning process during each week. This study indicated both usability issues and barriers when using Jeliot 3. We could also analyze the different purposes (e.g. for coding, debugging or testing) in which Jeliot was used during the course. Furthermore, it explained the attitudes of students for adopting new tools in the middle of their learning process because Jeliot 3 was introduced to them only during the second programming course and practices and strategies for programming had already been formed. We are currently extending the work with similar methodology in order to understand how the attitudes develop and problems change during the first programming course. However, we understand that by only using questionnaires other aspects of learning cannot be properly investigated.

Lessons Learned from Questionnaire Studies

Questionnaire-based studies are known for a bias caused by subjective responses of participants. Questionnaires require validation with similar user population in order to be usable. They tend to give only shallow knowledge of the users actions, thoughts and reflections and should mainly be used as a secondary source of information. However, in some contexts such as in distance learning, they might be the only feasible way of collecting data from a larger group of users.

DISCUSSION AND CONCLUSIONS

The selection of proper experimental methodology is crucial in order to properly investigate the phenomena. In this paper, we presented our experiences with three empirical methodologies to study human behavior, interaction and learning with program visualization. Each of the approaches provides the research agenda with important source of data. Classroom studies inform about the practices taking place in this context and can generate testable hypotheses. Controlled experiments, when designed well, can provide answers to the previously established hypotheses and can give accurate insights into interaction and cognitive processes involved in programming. Data from surveys and questionnaire studies can be used both to collect data related to attitudes and current practices, and generate testable hypotheses. Furthermore, all these methods can indicate issues for further development in the form of usability problems or unexpected behavior of users.

Each of the described methodologies has its own place in the research and development cycle. From our extensive experience, many types of methodologies to study interaction and learning with visualizations have to be applied to triangulate the findings. While the studies using controlled experiments prevailed in the past, classroom longitudinal research starts to appear. We think that both short-term and longitudinal studies are needed as well as quantitative, qualitative and especially mixed methods studies.

References

- Bednarik, R. (2005) The Technology, Usability & Pedagogy (TUP) online service. <http://cs.joensuu.fi/~tup>. Accessed 15.1.2006.
- Bednarik, R., Gert, P., Miraftebi, R., Tukiainen, M. (2004). Development of the TUP Model - Evaluating Educational Software. In Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies (ICALT 2004), Joensuu, Finland, August 30 - September 1, 2004, IEEE Computer Society, pp. 699-701.
- Bednarik, R., Myller, N. (2006). Predicting comprehension outcomes from eye-movement data. Manuscript submitted.
- Bednarik, R., Myller, N., Sutinen, E., Tukiainen, M. (2006). Analyzing Individual Differences in Program Comprehension. To appear in *Technology, Instruction, Cognition and Learning (TICL)*, 3(1), special issue on modeling and simulation.
- Ben-Ari, M., Myller, N., Sutinen, E., Tarhio, J. (2002). Perspectives on Program Animation with Jeliot. In S. Diehl (Ed.), *Software Visualization* (pp. 31–45), Berlin: Springer-Verlag.
- Ben-Bassat Levy, R., Ben-Ari, M., Uronen, P.A. (2003). The Jeliot 2000 Program Animation System. *Computers & Education*, 40(1), 1–15.
- Chi, M. T. H. (1997). Quantifying qualitative analyses of verbal data: a practical guide. *Journal of Learning Sciences* 6(3), pp. 271 – 315.
- 42 Kannusmäki, O., Moreno, A., Myller, N., and Sutinen, E. (2004) What a Novice Wants: Students Using Program Visualization in Distance Programming Course. In A. Korhonen (Ed.), *Proceedings of the Third Program Visualization Workshop (PVW 2004)*, Research Report CS-RR-407, Department of Computer Science, University of Warwick, UK. pp. 126–133.
- Lattu, M., Meisalo, V., Tarhio, J., 2003. A visualization tool as a demonstration aid. *Computers & Education* 41 (2), 133–148.
- Lattu, M., Tarhio, J., Meisalo, V., 2000. How a Visualization Tool Can Be Used - Evaluating a Tool in a Research & Development Project. In: *12th Workshop of the Psychology of Programming Interest Group*. Corenza, Italy, pp. 19– 32.
- Mayer R. E. (1981) *The Psychology of How Novices Learn Computer Programming*. *ACM Computing Surveys (CSUR)* 13(1), pp. 121 – 141.
- Moreno, A., Myller, N., Sutinen, E., Ben-Ari, M. (2004). Visualizing Programs with Jeliot 3. In *Proceedings of Advanced Visual Interfaces, AVI 2004* (pp. 373–376).
- Nevalainen S., Sajaniemi J. (2005) Short-Term Effects of Graphical versus Textual Visualisation of Variables on Program Perception. P. Romero, J. Good, S. Bryant, E. A. Chaparro (eds.) *Proceedings of the 17th Annual Workshop of the Psychology of Programming Interest Group (PPIG 2005)*. University of Sussex, U.K., 77-91.
- Stasko, J., T., Hundhausen, C., D. (2005). Algorithm Visualization. In Fincher, S., Petre, M. (Eds.), *Computer Science Education Research*, pp. 199-228.