UNIVERSITY OF JOENSUU

COMPUTER SCIENCE AND STATISTICS

DISSERTATIONS 23

Niko Myller

Collaborative Software Visualization for Learning: Theory and Applications

Academic dissertation

To be presented, with the permission of the Faculty of Science of the University of Joensuu, for public criticism at SciFest auditorium Space in Joensuu arena, Mehtimäenaukio 2, Joensuu, on April 17th 2009, at 12 noon.

UNIVERSITY OF JOENSUU

2009

Supervisors Professor Erkki Sutinen
Department of Computer Science and Statistics
University of Joensuu
Joensuu, FINLAND

Associate Professor Mordechai Ben-Ari
Department of Science Teaching
Weizmann Institute of Science
Rehovot, ISRAEL

Reviewers Professor Jari Lavonen
Department of Applied Sciences of Education
University of Helsinki
Helsinki, FINLAND

Associate Professor Michael E. Caspersen
Department of Computer Science
University of Aarhus
Aarhus, DENMARK

Opponent Associate Professor Christopher D. Hundhausen
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA, USA

ISBN 978-952-219-233-2 (printed)
ISBN 978-952-219-234-9 (PDF)
ISSN 1796-8100 (printed)
ISSN 1796-8119 (PDF)
Computing Reviews (1998) Classification: K.3.2, H.5.1, I.6.8

Joensuun yliopistopaino
Joensuu 2009

Collaborative Software Visualization for Learning: Theory and Applications

Niko Myller

Department of Computer Science and Statistics
University of Joensuu
P.O.Box 111, FIN-80101 Joensuu, FINLAND
`niko.myller@cs.joensuu.fi`

University of Joensuu, Computer Science, Dissertations 23
Joensuu, 2009, 183 pages
ISBN 978-952-219-233-2 (printed), 978-952-219-234-9 (PDF)
ISSN 1796-8100 (printed), 1796-8119 (PDF)

Abstract

As collaborative learning in general, and pair programming in particular, has become widely adopted in computer science education, so has the use of pedagogical visualization tools for facilitating collaboration. This introduces new challenges to the visualization tools, and thus, research and theory to support the development of collaborative visualization tools is needed. Currently, there is little theory on collaborative learning with visualizations, and few studies on their effect on each other.

In the research reported in this thesis, the collaborative use of software visualizations has been studied. The research is based on empirical studies of students who are learning programming or data structures and algorithms. The focus of the studies has been on both the collaborative learning process and the learning outcomes. The engagement taxonomy of visualizations has been extended in order to classify finer variations of the engagement resulting from the use of the visualization tools. The empirical studies have been carried out to analyze the collaborative learning process and outcomes in three different institutions and they have utilized three different visualization tools, namely Jeliot 3, TRAKLA2, and BlueJ. The hypotheses that were formed during the research are that the increase in the engagement level between the learners and the visualization tool results into more collaboration and better collaboration process (i.e., students interact more with each other and concentrate on the learning activities) and this, in turn, is hypothesized to increase the learning results of the students.

The studies were carried out in introductory programming, and data structures and algorithms courses. Students were working in pairs or small groups and their learning processes were recorded by using video cameras, screen capturing and audio recording. This material was analyzed using a video analysis where samples of the videos were categorized by the used engagement level, activities and discussion contents. The learning results were tested by using pre- and post-tests with between-subject design comparing different engagement levels and the quantitative results were analysed by using statistical methods. The studies have confirmed the hypotheses and the results are similar regardless of the tool or the institution.

Based on these results, I have created new ways to increase the engagement of the students during the viewing of visualization by creating automatic prediction question generation to Jeliot 3 program visualization tool. Furthermore, I have combined a collaborative authoring tool called Woven Stories with Jeliot 3 in order to form a new tool called JeCo, Jeliot Collaboratively.

In summary, the work reported in the thesis extends the existing theory with Extended Engagement Taxonomy and the hypotheses related to its applicability to collaborative use of visualizations, empirically confirms the hypotheses. By applying the theory, existing visualization tool is extended in order to support collaborative learning with visualizations in face-to-face and distance learning settings.

Keywords: Software visualization, Program visualizations, Algorith visualization,
Novice programmers, Data structures, Algorithms, Collaborative learning,
Computer-supported collaborative learning

Acknowledgements

During my doctoral research, I have been able to meet several people from all over the world, who have one way or the other influenced my thoughts and this research. I can only mention here a few and thus I also want to thank all the rest who I do not mention by name.

I consider myself lucky for being supervised by Professor Erkki Sutinen and Professor Mordechai "Moti" Ben-Ari from Weizmann Institute of Science. I have had the pleasure to have many long discussions with Erkki about the thesis and life in general. Those discussions have given me new ideas and hope in the struggle to get the thesis done. I hope these discussions have been a learning experience for both of us. Erkki has also aided me to become invited to other universities and thus given me an opportunity to experience the world from different points of view. Moti has been an invaluable source of help in the design of my research and in the analysis and reporting of the results. He has also encouraged me to go on while I have been hesitating. I am thankful to Moti that he hosted me at the Weizmann Institute of Science in the beginning of my doctoral studies. During this visit, we were able to lay the ground work for my doctoral research.

I am very happy that I have been able to work on Jeliot 3 with Andrés Moreno and Roman Bednarik as well as Ronit Ben-Bassat Levy from Weizmann Institute of Science. It has been a long but interesting journey from the autumn 2002 when we started to work on Jeliot 3 with Andrés. Andrés and Roman have given me helpful feedback on my research and supported me also outside the work as well as been my co-authors in several publications. Ronit is a tireless fan of Jeliot 3 and her use of and research on Jeliot has inspired me to work on Jeliot and research its use

Ari Korhonen and Mikko-Jussi Laakso have been my research collaborators at Helsinki University of Technology and University of Turku, respectively. With Ari and Mikko, we have carried out studies that are reported in several publications in this thesis. I am glad that Ari and Mikko were open to the idea of joint collaboration between researchers from three universities and that we could run experiments on their courses.

I thank Jussi Nuutinen for being my co-author in a paper presented in this thesis. Jussi has also brought joy into my working days at the university with the short gaming sessions.

It has been also a pleasure to work with Tuomo Kakkonen on automated essay grading and information retrieval. Although this research has not been directly related to my thesis, it has enabled me to see outside the box and look at my research from new perspectives.

I would like to thank Professor Jari Lavonen from the Department of Applied Sciences of Education at the University of Helsinki and Professor Michael E. Caspersen from the Department of Computer Science at the University of Aarhus for being my pre-examiners and giving valuable comments and guidance for this and future research. I am honored and excited to be able to have Professor Christopher D. Hundhausen from the School of Electrical Engineering and Computer Science at Washington State University as my opponent.

I am grateful to the East Finland Graduate School in Computer Science and Engineering (ECSE) and Centre for International Mobility (CIMO) for their financial support. I am also thankful to the Department of Computer Science and Statistics at the University of Joensuu for all the resources, material and immaterial, that I have been able to use or received during my research and studies, and all the colleagues that have been helping me along the way. Parts of this research have been conducted while I have been a visiting researcher at Massey University (Palmerston North, New Zealand) hosted by Professor Kinshuk and a visiting lecturer at Iringa University College of Tumaini University (Iringa, Tanzania).

My thanks also go to the instructors and students of all the courses, from which I have been able to collect materials for my research, at the University of Joensuu, University of Turku and Helsinki University of Technology.

My parents and family have given me all the ingredients for a good and happy life and values that I can be proud of. They have been guiding me and giving me support on those moments when I have been in doubt if I ever get to the finish line with my thesis. I extend these thanks also to my whole extended family who have been supporting me throughout this journey.

Last but foremost, I thank my significant other, Sini, without whom I would be incomplete and would not have been able to finish this work. Thank you for putting up with me, even though I spent sometimes days and nights with the thesis and not with you. I am grateful that I can share my life and research with you, and that you are willing to take risks like going to live in Tanzania with me.

Espoo, April 2009
- Niko Myller

List of Original Publications

P1. Myller, N., Bednarik, R., Ben-Ari, M., and Sutinen, E. (2009). Extending the engagement taxonomy: software visualization and collaborative learning. *The ACM Transactions on Computing Education*, 9(1), Article 7.

P2. Myller, N., Laakso, M., and Korhonen, A. (2007). Analyzing engagement taxonomy in collaborative algorithm visualization. In Hughes, J., Peiris, D. R., and Tymann, P. T., editors, *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE '07)*, pages 251–255, New York, NY, USA. ACM Press.

P3. Laakso, M.-J., Myller, N., and Korhonen, A. (2009). Comparing learning performance of students using algorithm visualizations collaboratively on different engagement levels. Accepted for publication in *Journal of Educational Technology & Society.*

P4. Korhonen, A., Laakso, M.-J., and Myller, N. (2009). How does algorithm visualization affect collaboration? Video Analysis of Engagement and Discussions. In Proceedings of the *5th International Conference on Web Information Systems and Technologies (WEBIST)*, pages 479–488.

P5. Myller, N. (2007). Automatic generation of prediction questions during program visualization. *Electronic Notes in Theoretical Computer Science*, 178:43–49. (Proceedings of the Fourth Program Visualization Workshop).

P6. Myller, N. and Nuutinen, J. (2006). JeCo: Combining program visualization and story weaving. *Informatics in Education*, 5(2):195–206.

Contents

Chapter 1

Introduction

"There is nothing to do with computers that merits a Ph.D."
– Max Newman (a.k.a. Maxwell Neumann)

During the last few decades, several algorithm and program visualization (or in general software visualization (SV)) tools and techniques have been developed (e.g., Baecker, 1981, Roman et al., 1992, Boroni et al., 1996, Lahtinen et al., 1998, Pierson and Rodger, 1998, Crescenzi et al., 2000, Naps et al., 2000, Rößling and Freisleben, 2002, Sutinen et al., 2003, Kölling et al., 2003, Ben-Bassat Levy et al., 2003, Korhonen et al., 2004, Moreno et al., 2004a, Hundhausen and Brown, 2007a). However, the research in this area was for a long time concerned with the effects of the medium and technology, i.e., computer screen and animation, and how they should be produced and configured in order to enhance learning (Naps, 2005). The evaluations of the tools from this period received mixed results both supporting and rejecting the use of software visualizations for learning (Hundhausen et al., 2002) and it has been difficult to develop a theoretical basis for the reasons why a visualization tool or its use in education is successful or fails. Hundhausen et al. (2002) proceeded into this direction in their meta-study and found that one of the key features leading to successful learning results is the involvement or engagement in which students are committed during learning with the visualization (e.g., watching vs. interacting). This observation was based on the analysis of previous studies, which showed that those studies that compared different levels of student engagement during a visualization were much more likely to find a difference in learning outcomes in favor of the visualization tool. This work led to the development of *Engagement Taxonomy* (ET) (Naps et al., 2002), which describes different forms of engagement. It rests on the idea that higher engagement between a learner and a visualization results in better learning results. Based on the studies analyzed by Hundhausen et al. (2002) and Naps et al. (2002) and more recent studies (e.g., Naps and Grissom, 2002, Grissom et al., 2003, Urquiza-Fuentes and Velázquez-Iturbide, 2007), there seems to a consensus that the engagement that a visualization promotes is a fairly reliable indicator of its educational effectiveness. However, most of the studies testing the educational effectiveness of visualizations have been performed on individual learners and there is only little evidence that the same or similar results apply when students are learning in collaboration with the visualizations (Hundhausen, 2002, Hübscher-Younger and Narayanan, 2003, Hundhausen and Brown, 2008). The analysis of the collaborative learning with visualizations seems to be a logical next step in the progress of the research into the educational effectiveness of algorithm and program visualizations as also illustrated in Figure 1.1.

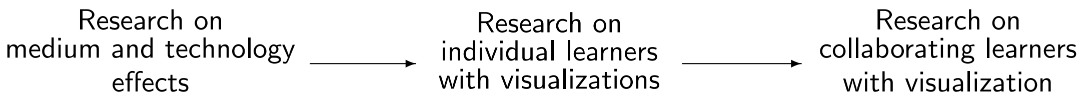

Research on medium and technology effects
Research on individual learners with visualizations
Research on collaborating learners with visualization

Figure 1.1: The progress of research on algorithm and program visualization.

As examples of the third phase of the research on software visualizations' educational effectiveness, Hundhausen (2002) and Hübscher-Younger and Narayanan (2003) have claimed that when communication and collaboration are accompanied with visualizations, students' learning of algorithms is enhanced. According to Hundhausen (2002), bi-directional communication between instructor and students supported by relevant visualizations helps students to learn algorithms because they get relevant feedback from their peers and instructors. Hübscher-Younger and Narayanan (2003) have shown that the construction, sharing and discussion of the students' representations of a certain algorithm aid students learn the algorithm better. However, very few studies exist that explicitly mention ET in the context of collaborative use of visualizations for learning (Hundhausen and Brown, 2008) and thus, there seems to be a missing link between ET and collaborative learning with visualizations.

When looking at the current situation from another perspective, the students are still struggling with and dropping out from their programming courses (McCracken et al., 2001, Robins et al., 2003, Lister et al., 2004, McGettrick et al., 2005, Bennedsen and Caspersen, 2007) and especially in the distance education courses on programming (Meisalo et al., 2003). The same seems to apply also to the data structures and algorithms courses (Korhonen et al., 2002, Milne and Rowe, 2002, Jain et al., 2006). Therefore, programming education has become one of the most active fields of research in the field of Computer Science Education (CSE). One solution that has been shown to help in these situations is the use of collaborative learning (Phillip et al., 1995, Wills et al., 1999, Chinn et al., 2007, Valdivia and Nussbaum, 2007, Beck and Chizhik, 2008, Teague and Roe, 2008) and pair programming (Nosek, 1998, McDowell et al., 2003, Nagappan et al., 2003, Williams et al., 2000). This also means that the visualization tools are used more often during collaborative learning or pair programming and this creates new demands on the visualization tools (Suthers and Hundhausen, 2003a, Bryant et al., 2005). However, theories or empirical results on how visualizations affect collaborative learning of programming or data structures and algorithms are rare.

A consequence of the shift from an individual learner using a visualization to collaborating learners using visualizations is that the learning process becomes more complex and more important, because the collaborating members need to consider also each other and make sure they have similar understanding in order to succeed. Visualization tools can also affect many aspects of the collaboration, for instance, resolution of misunderstandings or misconceptions (Roschelle, 1996), and it can serve as an external representation for the collaborating partners (Suthers and Hundhausen, 2003a). Moreover, Suthers and Hundhausen (2003a) have studied the effects of different kinds of visualizations and found that the structure and form of a visualization has an effect on what aspects of the information are discussed. Thus, there is a need not only to study the learning outcomes of a collaborative learning with visualizations but also the collaborative learning process leading to those results.

In this thesis, the issues related to the use of SV tools in collaborative learning are studied. I am especially interested in how the visualizations, particulary software visualizations, affect not only the collaborative learning *process* but also its *outcomes*. I build on the work from the research related to software visualizations and computer-supported collaborative learning (CSCL), and extend the existing theories and tools to support the teaching and learning of programming and data structures and algorithms.

1.1 Research Questions

The research reported in this thesis analyzes the effects of SV on the collaborative learning process and its outcomes in the context of introductory computer science education, especially in learning of programming and data structures and algorithms.

The motivation to study how to support collaborative learning in this context with visualizations is twofold. Firstly, previous research has shown that sharing and discussing algorithm representations have increased the students' learning outcomes, but research that analyzes the effects of visualizations on the learning process is also needed. Secondly, collaborative learning has been found to be a good way to motivate and support students' learning of introductory computer science, but there is very little empirically based knowledge that helps to understand this process and only a few tools to support it. Therefore, the first research question **RQ1** aims to study the effects of the SV tools on the collaborative learning process and the learning results.

> **RQ1:** *How do software visualization tools affect the collaborative learning process and its outcomes in the context of programming, and data structures and algorithms?*

As this question is quite broad and difficult to answer as such, it has been refined and narrowed to make it answerable based on the previous literature and in empirical studies. Thus, **RQ1** was divided into three narrower questions **RQ1.1**, **RQ1.2** and **RQ1.3** based on the previous literature and the experience gathered during the research process.

> **RQ1.1:** *Which qualities of the software visualization tools have been reported to affect collaborative learning?*

> **RQ1.2:** *How are the SV tools used in the collaborative learning?*

> **RQ1.3:** *How does the level of engagement between the learners and the visualization affect the collaboration process and its outcomes?*

The aim is to analyze and answer **RQ1.1** based on the review of previous literature. **RQ1.2** is targeted to further analyze the use of SV tools in collaborative learning based on empirical data that has been collected when students were learning collaboratively with SV tools. **RQ1.3** was formed based on the literature review that analyzed the educational effectiveness of visualizations and their collaborative use, and the empirical evidence obtained during this research so it can be seen as one of the answers to **RQ1.1**. Answers to these questions will produce new knowledge on the use of visualizations in learning and especially in collaborative learning. This can then be used to inform the design and development of new SV tools or extension of existing SV tools in order to support the collaborative learning with visualizations. Thus, the second research question **RQ2** is constructive in nature and targets the extension of existing SV tools so that the findings from the research results obtained in the study of **RQ1.1**, **RQ1.2** and **RQ1.3** can be utilized.

RQ2: *How can the existing SV tools be extended in order to better support collaborative learning of introductory computer science?*

This question was also split into two smaller questions in order to design and develop tools for face-to-face and distance learning situations as formulated in **RQ2.1** and **RQ2.2**. The context for the further development of the existing SV tools is the program visualization tool Jeliot 3 (Moreno et al., 2004a,b) which I have been designing and developing (Myller, 2004).

RQ2.1: *How can the existing SV tools be extended in order to have better support for collaborative learning of introductory computer science in a face-to-face learning situation?*

RQ2.2: *How can the existing SV tools be extended in order to have better support for collaborative learning of introductory computer science in a distance learning situation?*

In summary, the research aims at the development and empirical validation of a theory on aspects of the visualization affecting the collaborative learning process and its outcomes, and applying that knowledge to the further development of existing SV tools. Table 1.1 shows the research questions and the sections and the publications in this thesis that answer those research questions.

Research Question	**Section in Thesis**	**Publications**
RQ1.1	2.1.2, 2.2	(**P1**, **P3**)
RQ1.2	4.2.1	(**P1**, **P3**, **P4**)
RQ1.3	4.1, 4.2	(**P1**, **P2**, **P3**, **P4**)
RQ2.1	4.3	(**P5**)
RQ2.2	4.3	(**P6**)

Table 1.1: Research questions and the sections and publications answering them.

1.2 Methodology

As the research in this thesis is interdisciplinary, methods from multiple fields have been used. The use of multiple methods (or mixed methods (Johnson and Onwuegbuzie, 2004)) is becoming more common in the fields of CSE and educational technology research (Lister, 2005, Fincher and Petre, 2004). On the one hand, I have applied methods from behavioral or social sciences such as experiments and causal-comparative or observational studies as the quantitative methods (Gall et al., 2006, McGuigan, 1996). On the other hand, I have used the analysis of observations from videos similarly to the interaction analysis approach proposed by Jordan and Henderson (1995) and to the verbal data analysis described by Chi (1997). Finally, I constructed new tools and techniques that can be seen as constructive research (Lukka, 2003) or artifacts-building (Järvinen, 2004a,b).

From a methodological point of view, my research questions also indicate that I should use multiple methods in order to answer them (Järvinen, 2004b). In order to answer **RQ1.1**, a literature review has been carried out to collect the previous research results in the fields of software visualization research and computer-supported collaborative learning, and to understand what aspects of the visualizations have been identified as affecting collaboration. The video material from the empirical studies has also been used to analyze what aspects of the visualizations affected the collaborative learning process (cf. **RQ1.1**) and how the visualizations were used (cf. **RQ1.2**) as reported in (**P1**, **P3**, **P4**).

When it was identified that engagement between the learners and the visualization potentially have an effect on the learning process and results, it was tested using quantitative methods. However, as pointed out by Lister (2005) and (Ben-Bassat Levy et al., 2003),results obtained using qualitative methods can be used to describe the quantitative results. Therefore, research question **RQ1.3** has been investigated by using both quantitative and qualitative methods, but the emphasis has been on quantitative methods.

As a quantitative method, I have been using observational studies and experiments with a single-factor, between-subjects design with pre- and post-tests; the engagement level as the between-subject factor with normally two levels (McGuigan, 1996, Gall et al., 2006). This kind of design is common and has been used in CSE and educational technology research (e.g., Grissom et al., 2003, Sajaniemi and Kuittinen, 2005, Hundhausen and Brown, 2008). Furthermore, to analyze the impact of engagement on the learning process, I have used causal-comparative studies where the engagement was the observed independent variable (Gall et al., 2006). This is a less common mode of study and has the limitation that it cannot be used for statistical inference, but only to show correlations between measured dependent and observed independent variables. The quantitative data of the learning performance of the students was obtained from pre- and post-tests, and categorized episodes of video data were used in the analyses concerning the collaborative learning process.

In the categorizations of the video materials and their analysis, I have used approaches similar to those used by Lavonen et al. (2002), Suthers and Hundhausen (2003a), Suthers et al. (2003b), and Hundhausen and Brown (2008). In the observational studies and video analyses, the classification schemes of the students or episodes were defined beforehand. They were based on previous research and previously used categorizations (e.g., Hundhausen and Brown, 2008), and adapted to the needs of the subsequent analyses. The reliability of the resulting categorizations of the episodes was also checked using two raters and appropriate statistical tests of the inter-rater reliability (Landis and Koch, 1977). The category counts were then used in the quantitative analysis. However, not all of the materials were statistically analyzed, but only used to describe the frequencies and occurrences of events and discussion contents qualitatively.

Table 1.2 summarizes the relations between research questions and methods as well as their appearance in the sections of this thesis and in the original publications.

Research Question	**Methods**	**Sections**	**Publications**
RQ1.1	Literature review,	2	(**P1**, **P3**)
	Qualitative analysis	4.2.1	(**P1**, **P3**)
RQ1.2	Literature review,	2	(**P1**, **P3**, **P4**)
	Qualitative analyses	4.2.1	(**P1**, **P3**, **P4**)
RQ1.3	Literature review,	2.1.2, 2.2.2, 2.2.3	(**P1**)
	Constructive research,	4.1	(**P1**, **P4**)
	Experimentation,	4.2.2	(**P2**, **P3**)
	Observational studies	4.2.1	(**P1**, **P3**, **P4**)
	Qualitative analyses	4.2.1	(**P1**, **P3**, **P4**)
RQ2.1	Constructive research	4.3	(**P5**)
RQ2.2	Constructive research	4.3	(**P6**)

Table 1.2: The relationship between methods and research questions, and in which publications and sections they were used and described.

1.3 Contexts of the Research

Currently, when the world is more connected than ever, no research happens in isolation, but in a context that affects the methods that are used as well as the attitudes towards the research, even though most of this research has been quantitative in nature.

The research reported in this thesis has been carried out at three Finnish universities, the University of Joensuu, the University of Turku, and Helsinki University of Technology, by using three different SV tools: Jeliot 3, BlueJ, and TRAKLA2. Furthermore, parts of the analysis and development work reported in the thesis have been done while I was a visiting researcher or lecturer at three other universities, Weizmann Institute of Science in Israel, Massey University in New Zealand and Tumaini University in Tanzania. Thus, there have been multiple contexts and research groups that have been involved and have affected the studies and their results, but this should make the research results more reliable because similar kinds of results have been obtained in several contexts using three different SV tools and by working with several research groups and researchers.

From another point of view, I have been developing the program visualization tool Jeliot 3 for several years, and used it in teaching programming in face-to-face and online education; in addition, I have observed other teachers and students using it in various contexts. This has obviously had an effect on my ideas on how the visualization tools should be used in teaching and learning of computer science, as well as what aspects of the visualization I regard as useful or educationally effective. However, as a researcher I have tried to be in many ways both sensitive and open to new ideas and guided by the empirical data that I have analyzed and studied.

1.4 Organization of the Thesis

The thesis consists of seven Chapters and six publications, organized as follows. In Chapter 1, I motivate this work and explain the background that led to the research questions investigated in my research. I also describe the methodology that has been used to answer the research questions. In addition, the contexts of the research and the organization of the thesis are explained. In Chapter 2, I define the key terms that are used in this thesis and review the relevant literature that has guided the research work and is related to the reported studies. Chapter 3 summarizes the publications that are attached to this thesis and explains my contributions to those publications. Chapter 4 sums up the results that have been obtained during this research. The implications of the results are discussed in Chapter 5 as well as a the concluding remarks are given. The chapter concludes the introduction and summary part with future perspectives. The introduction and summary part of the thesis is followed by the six publications published in international peer-reviewed conferences and journals: (**P1**)–(**P6**).

Chapter 2

Literature Review

"A scientific truth does not triumph by convincing its opponents and making them see the light, but rather because its opponents eventually die and a new generation grows up that is familiar with it."
– Max Planck

In order to put my work into context, I will review literature related to this research. The literature review highlights those articles and studies that have influenced my own research or are related and relevant to the research questions and results of the studies reported in the thesis. Furthermore, I define the terminology used in this thesis based on the definitions in the literature and on my own use of the terms.

2.1 Software Visualization

Price et al. (1993) define *software visualization* (SV) as a combination of all the visualizations tools, technologies and techniques that have been made to help or to teach the process of software engineering (e.g., designing, comprehending or debugging a program). One of the reasons to build visualizations is to manage the complexity and emphasize those underlying relationships that might not be evident otherwise. SV uses different visual means such as, typography, graphics, and animation together with human-computer interaction to allow the construction, understanding and learning of computer software. This definition leaves the field very open and interdisciplinary, and thus, several subfields have emerged. In this thesis, software visualizations from three subfields, namely, program visualization, algorithm visualization and visual algorithm simulation, are used and studied. Although algorithm and program visualization are sometimes used interchangeably in the literature, they are as different as a program and an algorithm, i.e., a program is an actual implementation of an algorithm or algorithms. In this thesis, SV is also used to mean all the subfields that are covered in this thesis, namely program and algorithm visualization, and visual algorithm simulation.

Program visualization (PV) can be described as depicting the source code or the state of a program or its execution with the visual means. On the one hand, syntax highlighting can be regarded as one of the simplest forms of PV. On the other hand, the view in a program debugger or an animation of the execution of a program are more complex PVs. Nevertheless, PV is always linked to the underlying implementation of a program that the PV is illustrating. Due to this linkage and the lower abstraction level, PVs are often easier to generate (semi-)automatically compared to algorithm visualizations, which increases the ease with which they can be used, but it makes them sometimes difficult to understand, because the visualizations often show low level details and it is hard to automatically select the important parts of the program that should be visualized and what can be left out.

Algorithm visualization (AV) can be seen as illustrations or animations of data structures and their operations according to a certain algorithm. For instance, an algorithm animation could depict all the states of a data structure, as well as the transitions between those states during the execution of the algorithm. AVs are often abstractions of the underlying data structures and algorithm (or their implementation), so that only the essential parts for understanding are shown. This makes the automatic generation of AVs difficult but increases their understandability, because the visualization concentrates only on the necessary parts and shows them on higher abstraction level.

Visual algorithm simulation (VAS) (Korhonen, 2003) is a special case of AV in which the execution path of an algorithm is searched when the data structures, their initial and terminating states, and the permitted intermediate states are given. Thus, the simulation refers to the exploration of the possible execution paths and finding of the path that matches the given algorithm in the given settings. For example, the search for the correct order to visit nodes in a weighted directed graph with a depth-first search algorithm (which selects the next visited node by taking the lowest-weighted edge to an unvisited node when the starting node is given by using a visualization of the graph) can be seen as an visual algorithm simulation.

2.1.1 Visualization Tools Used in This Research

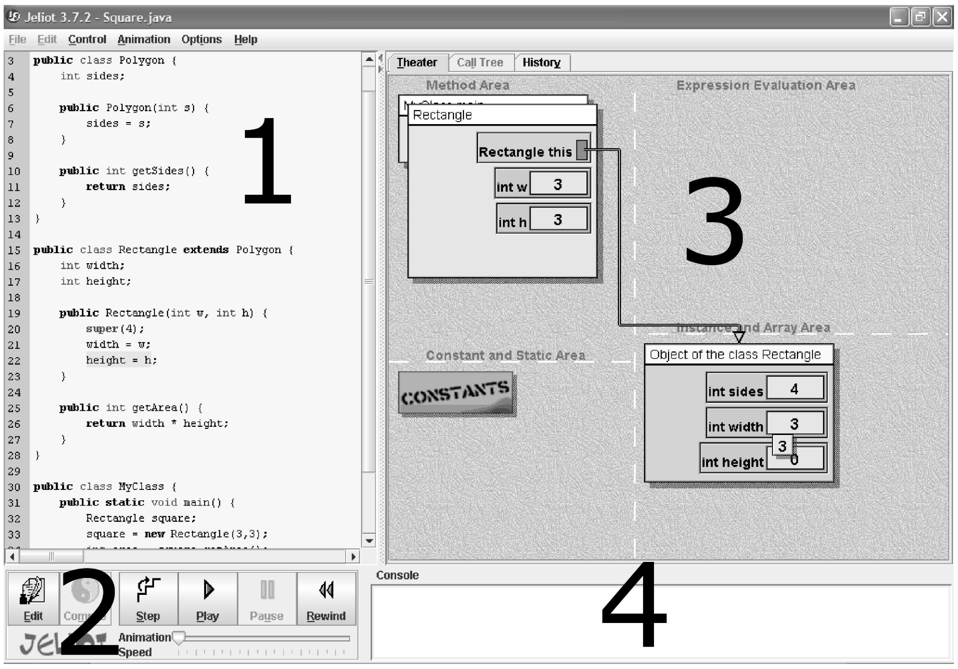
During my research, I have studied the effects of visualizations by using three different SV tools, namely Jeliot 3, BlueJ and TRAKLA2. One of the reasons for using multiple tools was to increase the credibility to the results by showing that the results are tool independent. In this section, I introduce the tools that were used as research instruments, explain their background and describe empirical studies carried out on the use of the tools.

Jeliot 3

Jeliot 3 (Moreno et al., 2004a,b) is a PV tool that is targeted especially to novice learners of Java programming. The development and research on Jeliot (Ben-Ari et al., 2002) has produced three previous versions, called Eliot (Lahtinen et al., 1998), Jeliot I (Sutinen et al., 2003), and Jeliot 2000 (Ben-Bassat Levy et al., 2003). Jeliot 3 is a generalization of its predecessor Jeliot 2000; the Java language coverage of Jeliot 2000 was extended by replacing the hand-made Java interpreter in Jeliot 2000 with a full-fledged Java interpreter, DynamicJava (Hillion, 2002). Most of the user interface and visualization components were maintained from Jeliot 2000 and only extended to visualize the newly supported Java language features (Myller, 2004). DynamicJava and the visualization engine were connected to each other by using a program trace description language, MCode, which is language independent and designed for this purpose (Moreno, 2005).

The main features of Jeliot 3 are the automatic and complete visualization of Java programs within a novice-oriented user interface. Automatic visualization means that a user can write a Java program without any Jeliot 3 specific code and animate the program's execution with Jeliot 3 with just two mouse clicks. Complete means that Jeliot 3 visualizes all aspects of the program execution from expression evaluation to object allocation. The animation in Jeliot 3 can be seen as a high level abstraction of the behavior of the Java Virtual Machine during program execution.

The user interface of Jeliot 3 can be seen in Figure 2.1. The user interface is divided into four areas: the code editor (1), the control panel (2), the visualizations (3), and the output area (4). The code editor contains the source code of the program that is currently being edited or visualized. The control panel is used to control the visualizations, whether it is playing, stepping, pausing or rewinding the animation. The visualizations area currently contains two different kinds of visualizations, namely the theater and the call tree. In addition, there is a history view, which shows the previous states of the animation in the theater as a series of snapshot images. The theater is the main visualization in Jeliot 3 and it shows all aspects of the program execution. The theater can be further divided into four areas: method area, expression evaluation area, constant and static area, and instance and array area. The method area shows the method frames that are currently active in the execution stack and the local variables stored in each of the method frames. All the expression evaluation takes place in the expression evaluation area. The constant and static area contains the static class variables as well as constant box from which the literal constants appear. The instance and array area is like the heap in the Java virtual machine and it stores all the instances that are currently referenced by the program. The output area shows the output of the current program.



```
public class Polygon {
    int sides;

    public Polygon(int s) {
        sides = s;
    }

    public int getSides() {
        return sides;
    }
}

public class Rectangle extends Polygon {
    int width;
    int height;

    public Rectangle(int w, int h) {
        super(4);
        width = w;
        height = h;
    }

    public int getArea() {
        return width * height;
    }
}

public class MyClass {
    public static void main() {
        Rectangle square;
        square = new Rectangle(3,3);
```

Jeliot 3.7.2 - Square.java
File Edit Control Animation Options Help
1
2
3
4
Theater Call Tree History
Method Area
Expression Evaluation Area
Rectangle
Rectangle this
int w 3
int h 3
Instance and Array Area
Constant and Static Area
CONSTANTS
Object of the class Rectangle
int sides 4
int width 3
int height 0
3
Console
Edit Compile Step Play Pause Rewind
Animation Speed

Figure 2.1: The user interface of Jeliot 3.

Jeliot 3 was designed for two different purposes. Firstly, it is a tool for the teacher to explain the meaning and behavior of different language constructs and to do what-if-type analyses in the classroom together with the students. Secondly, Jeliot 3 can be used by the students for independent study, including development and testing of their course assignments. As Jeliot 3 is designed in a modular way to be extensible, and it is published as an open-source project under GPL, several projects have used Jeliot 3 as a visualization component or developed new kinds of visualization tools based on Jeliot 3 (Bednarik et al., 2006c).

The development of different versions of Jeliot has been research-oriented and every version of Jeliot has been evaluated in empirical experiments. The results of those experiments have then guided the design of the next version of Jeliot. The first version of the Jeliot family, Eliot, was developed in order to simplify the development of algorithm visualizations. The studies done by Sutinen et al. (1997) and Markkanen et al. (1998) on the use of Eliot in a classroom indicated that the visualization tool can increase the motivation of the students and the quality of the program code and its documentation. This showed that the concept of program visualization in the way it was done in Eliot was feasible and worth further study. However, due to the restrictions of the execution environment of Eliot (it could be run only on certain Unix machines), it was replaced with Jeliot I, which was implemented in Java and could be run in an Internet browser.

The observation studies done to analyze the usage of Jeliot I by Lattu et al. (2000) and Lattu et al. (2003) described the different utilizations of the visualization tool during programming courses. They found five different use cases of Jeliot: supporting the presentation of new concepts, explaining an example program to students, tutoring the students during independent work, completion of programming exercises and independent study outside of the classroom. It was also found that true novices had difficulties in using and understanding Jeliot I because the visualization was coarse so that it showed only the changes in the variables, but the control flow of a program was difficult to understand. Furthermore, the large number of options that enabled the flexible semi-automatic generation of the animation was overwhelming. Thus, it was decided that in Jeliot 2000 the animation generation was to be fully automatic and complete, showing the full control and data flow, and that the user interface was to be oriented to the novices.

In a classroom experiment, Ben-Bassat Levy et al. (2003) evaluated the use of Jeliot 2000 in the teaching and learning of programming. The results showed that Jeliot 2000 especially helped mediocre students to perform better and to acquire a mental model as well as vocabulary that enabled them to explain situations that they had not experienced before. However, the language coverage of Jeliot 2000 was very limited as it did not cover object-oriented features of Java language and so these were included into Jeliot 3.

Kannusmäki et al. (2004) collected students' opinions on the use of Jeliot 3 during a web-based distance education course on programming. The data was collected through open-ended questions, which students answered every two weeks during the course. In the analysis, students were classified into three categories, i.e., strong, mediocre and weak, according to their performance in the course. Based on the comments, it was found that in this context the weak students benefited most from the visualizations and the strong students thought that they did not need this kind of tool anymore. This has led to the idea of developing adaptive program visualization capabilities into Jeliot (Moreno et al., 2007a). The idea is that students and teachers could either manually select which parts of the visualization they are interested in, or Jeliot could create a user model on the topics that the student has already learned and adapt the visualization according to the model.

Another study using qualitative methods to describe the use of Jeliot 3 is that of Moreno and Joy (2007). Students' use of Jeliot 3 was evaluated in the context of a programming course and its laboratory sessions. The progress of six students was followed and analyzed through interviews. In the data analysis, it was found that students used the tool in two modes: as a learning aid and as a debugger.

Students explained that the use of Jeliot 3 was especially beneficial when studying the concepts related to objects and arrays. However, most of the students could not explain the process of object creation in the interviews. They used it also to debug the programming assignments, as well as the programming project. However, they noted that viewing the visualization should be optional.

A series of studies that tracked the eye-movements of Jeliot 3 users has been performed (Bednarik et al., 2006a,b, 2005a,b). In these studies, the usage of Jeliot 3 during a program comprehension and debugging was analyzed. It was found that Jeliot 3 was used and visually attended differently depending on the programming experience of the user. Furthermore, the influence of the attended areas in the visualization on program comprehension was analyzed and some weak correlations were found. These studies have resulted in the design of a revised user interface and visualization in Jeliot, but it has not yet been implemented.

Another research direction of Jeliot 3 is the creation and use of conflictive animation (Moreno et al., 2007b). The conflictive animations can contain errors and the students need to spot these errors from the visualization and report them. This can help the learning of programming concepts in two ways. Firstly, this kind of game-like mode of visualization can engage students to view and interact with the visualization, and, secondly, students can test and train their understanding of the various concepts of programming.

The effects of visualization in Jeliot 3 on the learning process have also been researched, although not in a collaborative environment. In a classroom study, Ebel and Ben-Ari (2006) showed that program visualization of Jeliot 3 increases the attention of students to the material being taught and minimizes the unattentive behavior.

Jeliot 3 was used in the empirical study reported in (**P1**) together with BlueJ (see Section 2.1.1). The use of these tools together was made possible with the integration between Jeliot 3 and BlueJ (Myller et al., 2007a). In addition to this, the description of why and how an automatic prediction-type question generation was added to Jeliot 3 is given in (**P5**), and the integration of Jeliot 3 and Woven Stories, a collaborative authoring tool, is described in (**P6**).

BlueJ

BlueJ (Kölling et al., 2003) is a novice-oriented integrated development environment (IDE) for Java. BlueJ is especially designed for teaching and learning of object-oriented programming. Its main view, which is shown in Figure 2.2, contains the class diagram (upper part) and the object bench (lower part). The class diagram is an abstract view of the structure of a program and it shows the classes and their relationships. It is possible to interact with the classes in the diagram through popup menus to execute the main method or to create an object on the object bench. The object bench allows the user to interact with the objects by calling methods on the object or by inspecting the values of the fields in the object. In this way, it is possible to create complex behavior of the objects without writing a line of code. BlueJ also provides a source code editor, a unit testing facility (Patterson et al., 2003) and support for distributed group work (Fisker et al., 2008). These features have made BlueJ one of the most widely used teaching and learning tools in programming courses.

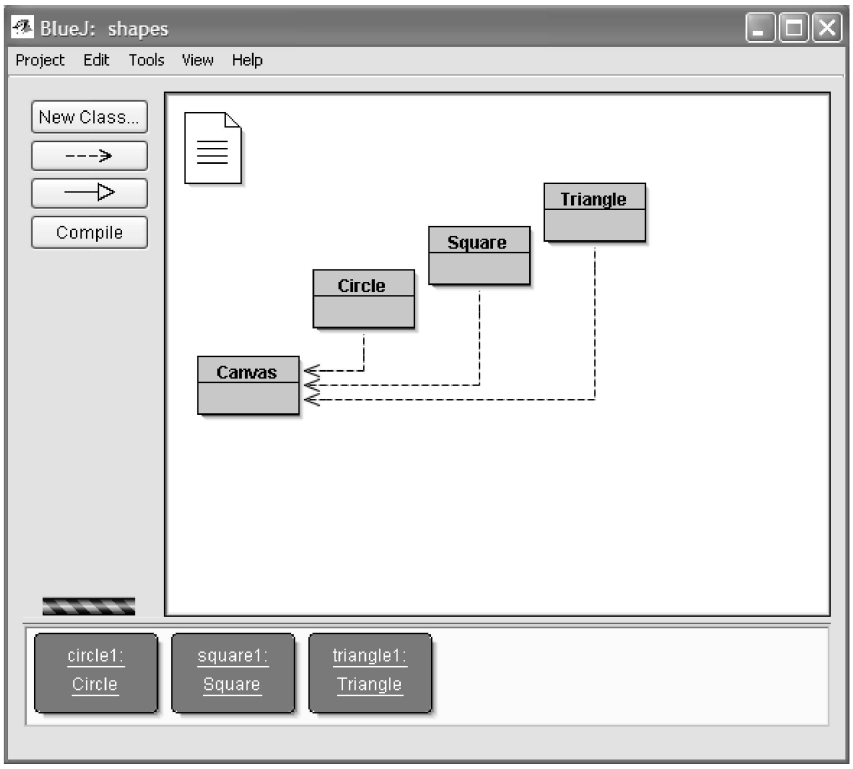


Figure 2.2: The user interface of BlueJ.

BlueJ provides extension interfaces that allow the development of plug-in components. There have been several extensions developed for BlueJ for different purposes, for example, support for design patterns (Paterson et al., 2006, Paterson and Haddow, 2007), user compilation behavior monitoring (Jadud, 2006), program visualization with Jeliot 3 (Myller et al., 2007a), and class- and object-relationship understanding (Paterson and Haddow, 2007).

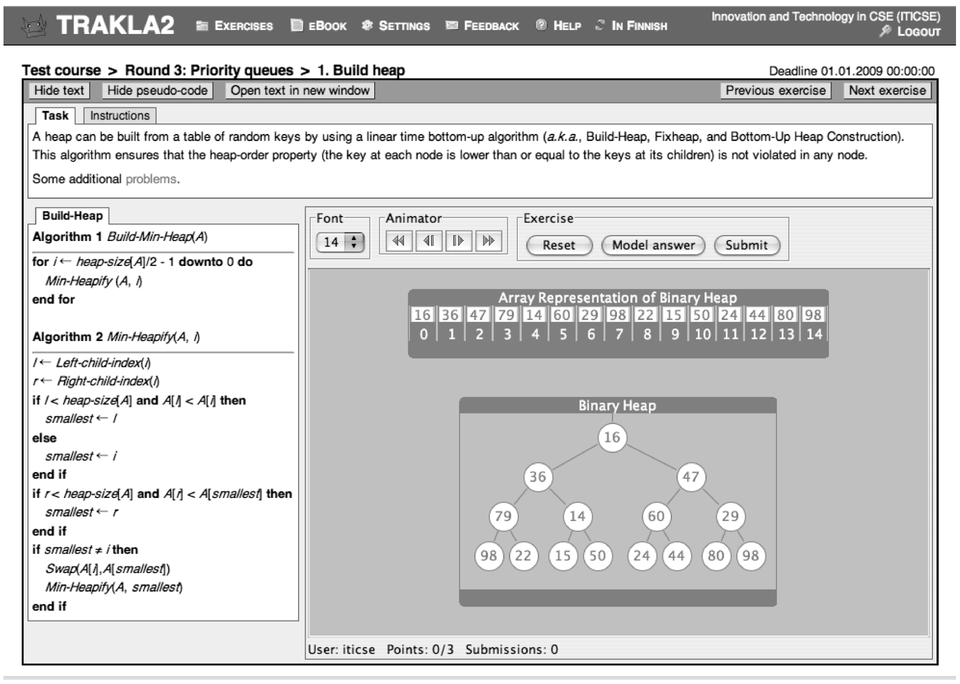
The educational use of BlueJ has been researched by Haaster and Hagan (2004), and Ragonis and Ben-Ari (2005a,b). Haaster and Hagan (2004) designed and carried out a survey on students opinions on the usability, the support for the programming paradigm, and the support for teaching and learning of BlueJ in an introductory course on programming. It was found that students were happy to use BlueJ and had only a few technical problems with it. They were also using various features of BlueJ and expected that those would have had a positive impact on their course performance. However, there is no clear evidence that the use of BlueJ actually resulted into better learning results. The study by Ragonis and Ben-Ari (2005a,b) identifies students' conceptions of object-oriented programming concepts. The study is not mainly concerned with BlueJ, but it was carried out in a classroom where BlueJ was used and some of the conceptions were related to the use of BlueJ. For example, students had difficulty understand the data and control flow of a program as these are not made salient in BlueJ. Students also struggled to develop programs that used a main method as it was not needed in BlueJ. The overall success of the students was still positive and some of that can be attributed to the use of BlueJ. Xinogalos et al. (2007) has also reported an increase in learning results when BlueJ has been used for a long period of time.

BlueJ was used as a programming environment for learning by the students in the study that was reported in (**P1**).

TRAKLA2

A visual algorithm simulation exercise environment, TRAKLA2 (Korhonen et al., 2004, Malmi et al., 2004), is based on the algorithm visualization and simulation framework Matrix (Korhonen et al., 2004). The key features of the environment are the direct manipulation of the data structure visualizations and the automatic assessment of the algorithm simulations. Furthermore, each student is provided with individual input data for the algorithm in order to avoid copying and to allow practicing of the algorithm simulation with different input data. This also allows multiple resubmission of the algorithm simulations exercises as there is no possibility for using a trial-and-error method to solve the exercise.

The user interface, which is illustrated in Figure 2.3, is built on the concept of direct manipulation that is implemented through dragging and dropping of visual components of the data structures. For instance, a user can drag tree nodes or array cells from one data structure to another or within a data structure (e.g., swap of elements in an array) and the algorithm simulation requires that these operations are done in a correct order according to an algorithm. A normal sequence of an algorithm simulation exercise begins when a user is given an input, an initial state of the data structure and the algorithm to simulate. The input can be, for example, an array of random keys that should be inserted into an empty binary search tree, or it could be a graph from which the minimum spanning tree should be found using Prim's algorithm. In Figure 2.3, the input is an array of keys and its binary tree representation and the algorithm is build-heap because the tree does not satisfy the heap property. The user is supposed to simulate the exercise by dragging and dropping the visual components in the order that simulates the given algorithm on the given data structures. If the user makes any mistakes or does not remember the previous steps of the simulation, s/he can go back to previous stages by pressing the backward arrow and s/he can then return to the latest state pressing the button with the forward arrow. When the user thinks that the algorithm on the data structures is simulated correctly, s/he can press the submit button and s/he receives instant feedback on the exercise. The response is automatically compared to the correct sequence of steps and the number of correct steps from all steps is shown to the user. Then, the user can view the model answer that is a step-by-step sequence of the correct steps of the algorithm simulation. After viewing the model answer, or submitting the exercise, the user needs to reset the exercise with new input data before s/he can submit again.



Test course > Round 3: Priority queues > 1. Build heap

Deadline 01.01.2009 00:00:00

A heap can be built from a table of random keys by using a linear time bottom-up algorithm (*a.k.a.*, Build-Heap, Fixheap, and Bottom-Up Heap Construction). This algorithm ensures that the heap-order property (the key at each node is lower than or equal to the keys at its children) is not violated in any node.

Some additional problems.

```
Algorithm 1 Build-Min-Heap(A)
for i ← heap-size[A]/2 - 1 downto 0 do
  Min-Heapify (A, i)
end for

Algorithm 2 Min-Heapify(A, i)
l ← Left-child-index(i)
r ← Right-child-index(i)
if l < heap-size[A] and A[l] < A[i] then
  smallest ← l
else
  smallest ← i
end if
if r < heap-size[A] and A[r] < A[smallest] then
  smallest ← r
end if
if smallest ≠ i then
  Swap(A[i],A[smallest])
  Min-Heapify(A, smallest)
end if
```

16	36	47	79	14	60	29	98	22	15	50	24	44	80	98
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

User: iticse Points: 0/3 Submissions: 0

Figure 2.3: Example of the TRAKLA2 algorithm simulation exercise on the build-heap algorithm.

Korhonen et al. (2002) studied the use of algorithm simulation exercises in a course on data structures and algorithms. They analyzed how the learning results differed if the exercises were done in classroom or on the web. The study used three groups of students who solved either algorithm simulation exercises on the web, algorithm simulation exercises in a classroom with a human tutor or algorithm design exercises in a classroom with a human tutor. The results showed that the course results were similar when students performed same exercises on the web as in the classroom. The students who were solving algorithm design exercises in the classroom performed better than the other two groups, but this was to be expected because those exercises were more demanding. However, this group also had the highest drop-out rate, which could indicate that only the better students could succeed with the more difficult exercises.

Laakso et al. (2005) carried out a multi-perspective study on the utilization of TRAKLA2 in data structures and algorithms education. The study showed that the students' attitudes towards web-based learning became more positive when they used TRAKLA2, and they indicated that the most preferable learning settings include assignments that are done in the web-based environment as well as assignments that are done in a classroom. The overall course exam results also increased compared to the previous years when TRAKLA2 was not used.

In order to understand students misconceptions better, Seppälä et al. (2006) analyzed students' answers to the algorithm simulation exercises related to the binary heap data structure. They identified several misconceptions in the student submissions for the build-heap algorithm and classified them into different categories. Based on this categorization, automatic guidance can be provided for the students whose submission contained a misconception. Although the heap data structure was used in the studies reported in (**P2**, **P3**, **P4**), this feature was not used in them.

In the studies that are reported in (**P2**), (**P3**) and (**P4**), the TRAKLA2 algorithm simulation exercises related to the binary heaps were used. The exercises were inserted into the web-based learning materials which students used to learn the concepts related to binary heaps. The algorithm simulation exercise of build-heap algorithm is shown in the Figure 2.3

2.1.2 Educational Effectiveness of Software Visualizations

A large number of studies testing the educational effectiveness of software visualization have been carried out with mixed results (Hundhausen et al., 2002). Until recently, the reasons why results have been inconclusive have been unclear. However, the work of Hundhausen et al. (2002) has shed new light on the matter and this has resulted in the development of the *Engagement Taxonomy (ET)* (Naps et al., 2002). ET describes different forms of engagement that can be promoted by a visualization tool and it provides testable hypotheses about how the engagement level of a visualization affects the learning outcomes. The central idea of the taxonomy is that *a higher-level engagement between a learner and the visualization results in* *better learning outcomes.* The ET consists of six levels of engagement and they are described in Table 2.1.

Table 2.1: The engagement taxonomy, shown also in (**P1**).

Level	Description
No viewing	There is no visualization to be viewed
Viewing	The visualization is only looked at without any other form of engagement
Responding	Learners are presented with questions related to the visualization
Changing	Modification of the visualization is allowed, for example, by varying the input data set
Constructing	Learners are expected to create their own visualization of a program or an algorithm
Presenting	Learners present visualizations to others for feedback and discussion

The engagement level *no viewing* means that there is no visualization in which to engage. However, there might be something else such as textual learning materials to look at. Based on the previous results, passive *viewing* of a visualization does not seem to increase the learning results when compared with the *no viewing* level (Hundhausen et al., 2002, Naps et al., 2002, Naps, 2005). The visualization should always be accompanied with an activating component which makes the use of the visualization meaningful and engaging, such as *responding*, *changing*, *constructing* or *presenting* a visualization. In light of current research, the taxonomy forms a three-level hierarchy: no engagement, passive engagement, and active engagement (Naps, 2005). However, the differences in learning results between the levels of active engagement are still an open question, although there have been some attempts to compare them (Lauer, 2008b).

The hypotheses related to the engagement taxonomy are also in line with research in educational psychology and multimedia learning (Mayer, 2001), where it has been found that interactivity in multimedia has a positive effect on learning outcomes (Evans and Gibbons, 2007, Mayer and Chandler, 2001, Moreno and Mayer, 2000). This effect was found especially on transfer questions that tested the ability to understand the learned materials.

Since its introduction, the ET has been used to guide the design and implementation of SV tools, and recent studies have validated its applicability with varying visualization tools (Urquiza-Fuentes and Velázquez-Iturbide, 2007, Grissom et al., 2003, Naps and Grissom, 2002). Nevertheless, there are still studies that have gotten mixed results (e.g., Lauer, 2006, Jarc et al., 2000, Rhodes et al., 2006), but these results are normally explainable by confounding factors. This shows that there is still a need for methodological work in order to find ways to control all the confounding factors in studies.

There are several findings, which make it questionable if visualizations on the *responding* level result in better learning results compared to visualizations on the *viewing* level (Jarc et al., 2000, Grissom et al., 2003, Lauer, 2006, Rhodes et al., 2006, Lauer, 2008b); this needs to be resolved by further experimentation and analysis of the body of research concerning the comparison of *viewing* and *responding*. However, studies have shown that the differences in the learning results when comparing the *no viewing* and *responding* levels have been statistically significant (Byrne et al., 1999, Grissom et al., 2003). This indicates that the ET and the hypotheses it provides should be still improved and developed in order to describe all the variations of engagement and their effects.

Based on an analysis of the recent results, Lauer (2008b) has proposed modifications to the ET. He suggested splitting of the *viewing* level into two levels based on the terminology used in (**P1**, **P3**): *viewing* and *controlled viewing* (his original terms were *passive viewing* and *active viewing* in Lauer (2008a)). Controlled viewing can be defined as viewing during which users are able to pause the visualization and step the visualization backwards and forward at their own pace. Another modification is the splitting of the *constructing* level into three separate levels: *simulating*, *hand-constructing* and *code-based constructing*. *Simulating* refers to visualization similar to the visual algorithm simulation exercises in TRAKLA2. *Hand-constructing* refers to creation of a visualization without coding the algorithm or program but using ready-made visual components to create a visualization. *Code-based constructing* refers to the creation of a visualization by coding it or at least starting the process from the code. These modifications are partially similar to the modifications and additions proposed in (**P1**), but they were developed independently and from different perspectives. However, the publications (**P1**, **P3**), as well as our mutual communication (Lauer and Myller, 2008), have partially influenced the work of Lauer (2008b) as noted in his paper. The relationship of the different extensions of the ET to each other and to the original ET will be further discussed in Sections 4.1 and 5.1.

There are two notable issues regarding the research that has been performed on the educational effectiveness of the SV tools. Firstly, most of these studies have concentrated on the effects of the visualizations on *learning outcomes*. Secondly, most of these studies have studied the individual learners. There are a few exceptions to these claims (e.g., Hundhausen and Brown, 2008, Hübscher-Younger and Narayanan, 2003, Jehng and Chan, 1998), and they will be discussed in Section 2.2.3. In the research reported in this thesis, we have extended the scope of this kind of research in order to investigate the impact of the engagement with visualization tools on the *learning process* in (**P1**, **P4**). Furthermore, this has been done in the settings of collaborative learning with SV tools (**P1**, **P2**, **P3**, **P4**).

Another important point, which is often overlooked in the development of the software visualization tools for learning as well as in the studies that test their educational effectiveness, is that the visualization displays are complex, and therefore, students need time to learn how to read and understand them (Petre, 1995). Furthermore, visualizations contain information that is often encoded into the locations and layout of the visual elements (i.e., *secondary notation* (Petre, 1995, Petre and Green, 1993)), which can make the interpretation even more difficult. Therefore, it is necessary that students become familiar with the visualization tools provided for them; otherwise one cannot be sure what caused the effects, or if the potential effects did not materialize because students could not understand the visualization in order to receive its maximal benefits. In the studies reported in (**P1**, **P2**, **P3**, **P4**), students had been using the visualization tools before and were already familiar with them.

2.2 Computer-Supported Collaborative Learning

Computer-Supported Collaborative Learning (CSCL) is a field of study concerned with the process of learning with computers in a group of mutually engaged peers. The field emerged about two decades ago (Scardamalia et al., 1989, Roschelle, 1992, Dillenbourg and Self, 1992, Lehtinen et al., 1999) as a new inter-disciplinary study overlapping cooperative learning (Slavin, 1995) and computer-supported cooperative/collaborative work (Grudin, 1994). From theoretical point of view, CSCL is grounded on the works of Mead (1977), Vygotsky (1978), Piaget (1980), Slavin (1995) on social constructivism, distributed cognition and cooperative learning. For an overview of the development of the CSCL see the review by Lehtinen et al. (1999).

2.2.1 Successful Computer-Supported Collaboration and Collaborative Learning Process

It is difficult to define what a *successful collaboration process* consists of (Mattessich et al., 2004). When the computer and learning are introduced to the process the definition becomes even harder. However, when studying how the collaboration is affected by the visualizations, there is a need at least to define those parts of the collaboration that could be affected by the visualization and whether the effects are positive or not.

An attempt to define what a successful computer-supported collaboration is can be found in the research of Meier et al. (2007). Based on an extensive review of the literature and on empirical data analysis, they synthesized a two-layer model with five aspects and nine dimensions that could be regarded as components of a successful computer-supported collaboration process. These components are shown in Table 2.2).

Aspect	Dimension
Communication	1) Sustaining mutual understanding
Communication	2) Dialogue management
Joint information processing	3) Information pooling
Joint information processing	4) Reaching consensus
Coordination	5) Task division
Coordination	6) Time management
Coordination	7) Technical coordination
Interpersonal relationship	8) Reciprocal interaction
Motivation	9) Individual task orientation

Table 2.2: Aspects and dimensions of a successful collaboration process (Meier et al., 2007), shown in (**P1**).

These dimensions and aspects were used in a rating scheme of successful collaboration, and their validity and reliability was evaluated with empirical data (Meier et al., 2007). It was found that there exists a positive correlation between the correctness and quality of the final results of the collaboration and the ratings on the dimensions describing the collaboration process (Spada et al., 2005, Meier et al., 2007). This confirms the validity of the scheme. However, the inter-rater reliabilities of the ratings were not high and this causes doubt on the usage of the scheme. Nevertheless, the dimensions and aspects can be regarded as indicative of the qualities of successful collaboration, but they are hard to detect and agree on when the collaboration is analyzed.

The importance of discussion and their contents in collaborative learning with computers have been studied by Teasley (1997). In the studies, Teasley (1997) found that the amount of discussion is positively correlated with successful collaboration. In particular, discussions that contain *transactive reasoning* are even stronger indicators. Transactive reasoning is discussion about one's own thinking process or discussion about one's understanding of the partners' thinking processes (Berkowitz and Gibbs, 1983). Transactive reasoning can be further classified into several independent categories as presented in Table 2.3. When collaboratively learning programming or data structures and algorithms with visualizations, transactive reasoning could be, for instance:

- discussion about how the visualization was understood by the different group members;

- discussion about the visual components of a data structure and their relation to the concepts of the data structure;
- predictions of the next steps of the visualization with justification;
- explanation why the particular line of source code contains an error and how the student found the error.

Table 2.3: Transactive reasoning categories adapted from Berkowitz and Gibbs (1983) and Teasley (1997), and used in the data analysis in (**P1**).

Category	Description
Prediction	A participant tries to predict what will happen next and justifies the prediction.
Feedback Request	A participant ensures that others understand or agree with his/her position
Paraphrase	A participant paraphrases a discourse of another student in order to demonstrate that s/he understands it.
Justification	A participant justifies his/her position or reasoning.
Juxtaposition	A participant explains the differences between the positions or reasoning of other students and his/her own.
Completion	A participant completes another student's reasoning, for example, by filling out an unfinished sentence.
Clarification	A participant explains his/her reasoning in order to ensure that others understand it.
Refinement	A participant elaborates or qualifies his/her position in order to defend it against criticism.
Extension	A participant elaborates on a previous discourse.
Criticism	A participant criticizes the reasoning or position of another student and explains the reason for the criticism.
Integration	A participant combines different views into one common statement.

Successful collaboration and collaborative learning require at least: *interaction* as indicated by dimensions 1, 2, 4 and 8 from Meier et al. (2007) as well as the research done by Teasley (1997), *coordination* as shown by dimensions 3, 5, 6 and 7 from Meier et al. (2007), and *motivation* as demonstrated by dimension 9 from Meier et al. (2007). There exists evidence in the literature that visualizations can affect the motivation of the students. For example, Ebel and Ben-Ari (2006) demonstrated positive affective effects of program visualizations (i.e., *increased attention*) and there is also anecdotal evidence in the literature about the increased *motivation* of students when visualizations are used (Naps et al., 2002, Naps and Grissom, 2002, Grissom et al., 2003, Urquiza-Fuentes and Velázquez-Iturbide, 2007). In addition, *coordination* of the collaboration or collaborative learning have been shown to be supported by augmenting visualizations (Janssen et al., 2007). In the studies reported in this thesis, the rating scheme developed by Meier et al. (2007) was not used as such, but the studies concentrated on the communication and *interaction* and how the use of a visualization tool affects them (**P1**, **P4**). The transactive reasoning categories were used in the study reported in (**P1**).

2.2.2 Research on the Use of Visualizations in Collaborative Learning

There have been a number of studies concerning CSCL with a learning tool and its educational effectiveness (Lehtinen et al., 1999); however, there are only a few studies analyzing the use and impact of a visualization in collaborative learning. One of the earliest ones is the research of Roschelle (1996). In his thesis work, he developed a software tool for studying Newtonian mechanics called the *Envisioning Machine (EM)* and studied its use in collaborative learning. In the EM, students can directly manipulate velocity and acceleration vectors in a simple diagram showing the movements of a ball. The goal is to adjust the initial speed so that the ball flies according to the given trajectory. The study was concerned with the learning process and outcomes of pairs of students when they were learning with the tool. When analyzing the learning process, Roschelle recognized that learning tools used in collaboration should be designed to support communication even at the expense of epistemic fidelity or correctness in the eyes of an expert, as long as the essential information is available. Roschelle (1996) gives a number of guidelines in order to achieve this goal and the final one is: "one should design activities which actively engage students in doing and encounter [sic] meaningful experiential feedback as a consequence of their actions" (p. 14). Scaife and Rogers (1996) have recognized the same issue when proposing that the analysis of the interaction between the external presentation and users is a key research area.

Another long term research initiative is that of Suthers et al. (1997) on a collaborative inquiry learning tool, Belvédère. They have studied different aspects of CSCL and one of them is a hypothesis about *representational guidance* (Suthers, 1999). Representational guidance refers to the differences between the layout and form of the representations and how they make some information more visible and hide some other information, so that they either are discussed extensively or not at all.

In a controlled experiment, Suthers and Hundhausen (2003a) compared the representational guidance effects of three different representations, matrices, graphs and text, in face-to-face collaborative inquiry. Students' collaboration processes and learning results were analyzed when they were collecting data, forming hypotheses and investigating their evidential relations by using the tool with one of the representations, in order to understand the cause and effect of the given situation. In the analysis, Suthers and Hundhausen (2003a) found differences in the discussions that students had during the collaboration process and these differences could be attributed to the qualities of the representation that was used. For example, students who used matrices to record the data and hypotheses, as well as their evidential relations, discussed more about whether or not there exists an evidential relation between a data point and a hypothesis compared to the other two groups. This was predicted from theory of representational guidance because the matrix made all the possible relationships between data and hypotheses more visible compared to the other two representations.

They have expanded this research to on-line collaboration and found that the learning situation affected the discussions related to the representations (Suthers et al., 2003b). However, the collaboration-process differences did not materialize into significant differences in the learning outcomes measured by post-tests in either of the experiments. This could be due, at least partially, to the restricted laboratory environment where the experiments were carried out.

Communicative Dimensions (CD) (Hundhausen, 2005) is a framework for analyzing visualizations as media for communicating and sharing knowledge and ideas between users. CD was inspired by the Cognitive Dimensions framework, which describes the interaction possibilities and barriers of software tools from the perspective of the user (Green and Petre, 1996). CD describes the aspects of visualization environments that have an effect on communication between its users by using six dimensions: *programming salience*, *provisionality*, *story content*, *modifiability*, *controllability*, *referencability*. *Programming salience* measures the amount of low-level "programming" that is needed in order to produce a visualization. The rationale is that if a large amount of work goes into the fine tuning of the lower level details of the visualization, the lower level details will be the main focus of the discussion rather than the overall picture provided by the visualization. A visualization has low *provisionality* if it looks like a final product, which does not leave any room for discussion or critique, but explains it all. High *provisionality* means that a visualization is easier to approach and talk about because it is not looking so finished and polished, and leaves room for further discussions and refinements. Visualization with *story contents* uses narrative elements to explain or highlight domain concepts. This may help users discuss the subject matter in story-specific terms, especially if users are novices and lack the domain-specific terminology. However, if the metaphor or story that is used is unsuitable to that particular situation, it may hinder the communication. Thus, this dimension's benefits are controversial and depend on the context and use case. High *modifiability* of a visualization enhances communication, because users can easily test and analyze different options by dynamically modifying the visualization. For instance, this supports "what-if" type of analyses during a presentation or a lecture. *Controllability* expresses how easy it is to control the execution of a visualization. For example, a visualization with high controllability allows the user to jump into any step of the visualization whereas low controllability means that the visualization can only be played forward or not controlled at all. *Referencability* describes the ease of referring to the visualization by either pointing or otherwise annotating its components. The CD dimension *provisionality* has a direct link to the epistemic fidelity in the work of Roschelle (1996) so that low epistemic fidelity most probably results in high provisionality and *vice versa*. In addition, the modifiability, controllability and referencability seem to have indirect links to the work of Roschelle (1996). All these dimensions are features of a visualization or its interaction support and can either hinder or foster the communication between users, and therefore, affect the collaboration process.

The works of Hundhausen (2005), Suthers and Hundhausen (2003a), Scaife and Rogers (1996) and Roschelle (1996) show that visualization and the kinds of interactions it promotes with the learners affect the process of collaborative learning, and these effects and their causes should be further studied as they have an effect on the learning outcomes as well.

2.2.3 Research on the Use of Software Visualization in Collaborative Learning

As discussed earlier in Section 2.1.2, a large number of SV tools has been developed and their educational effectiveness have been studied in empirical evaluations. However, it is still unclear how these tools can and should be used in collaboration and what the effects of the visualizations are when used in collaborative learning. Nevertheless, there have been few tools and studies that have attempted to describe and analyze these situations.

Jehng and Chan (1998) reported on a distributed visual learning environment to support collaborative learning of programming in LISP-LOGO. In the evaluation of the tool it was found that the collaborating students outperformed individual learners in program generation tasks and it did not matter if the collaborating students were co-located or not. However, all groups performed equally well in program evaluation and completion tasks. This indicates that collaborative learning with visualizations can be more beneficial compared to individual learning, but the benefits seem to materialize when the tasks are more demanding and challenging.

Lavonen et al. (2002) and Lavonen et al. (2003) have studied the collaborative use of an icon-oriented visual programming environment, *Empirica Control (EC)*, in teaching and learning of technology and programming in elementary school. They found that students are able to easily master the use of the tool and basic concepts in programming (Lavonen et al., 2003). Furthermore, it was found that when students were working with EC, they were truly collaborating. For instance, in 62% of the analyzed episodes the students were discussing and working together towards a common goal (Lavonen et al., 2002).

Hundhausen (2002) studied the use of AV tools in algorithm visualization construction and presentation in a data structures and algorithm course. The ethnographic field study indicated that the construction and presentation of the visualizations help students to learn if the visualizations can be easily constructed and those activities concentrate and engage the students in relevant interactions between the instructor and the peers. If the construction of the visualization requires much work that is not relevant from the point of view of the learning goals, the benefits of the use of visualizations are lost (cf. Hundhausen, 2005). Similar results were also obtained by Hübscher-Younger and Narayanan (2003), who developed a web-based system that allowed students to publish their own algorithm representations (text, pictures, animations, multimedia) and discuss them on the web. The results of the study showed that the active students who created and shared their visualization and commented on others achieved higher grades than the passive students who only viewed and commented on the visualizations made by other students.

One of the only studies that has combined the engagement taxonomy and collaborative learning is the work of Hundhausen and Brown (2008). Based on previous research (Hundhausen et al., 2002, Hundhausen, 2002) on the use of algorithm visualization systems in the learning of data structures and algorithms, Hundhausen and Brown (2007a) have developed a tool called ALVIS, which allows the the construction and interactive presentation of "low fidelity" algorithm visualizations. In the study, Hundhausen and Brown (2008) compared ALVIS to a text editor as a tool for writing programs, and then compared the use of ALVIS as a visualization construction and presentation tool to simple art supplies. Students worked in pairs and were asked to write an algorithm in the SALSA programming language supported by ALVIS, and then to construct a visualization of the same algorithm, and present it to the instructor and the other students. Based on an interaction analysis of the videotaped materials, it was found that the pairs who engaged in *viewing* and *constructing* by using the ALVIS environment concentrated more on the solution, spent less time unproductively, and needed less help from the teaching assistant compared to the pairs of students who used a text editor and engaged into *no viewing* and *constructing* with art supplies. Moreover, students using ALVIS developed better code than pairs who used only a text editor. Although one of the explaining factors could be that the text editor group did not have the same kind of tools to execute and test their programs, this is still an indication that engagement levels can have an effect on the collaboration process and its outcomes.

A few other visualization tools have been developed in order to support collaborative learning of programming and data structures and algorithms. An example of this kind of a tool is VorteX (Ratcliffe and Thomas, 2004, Ridgway et al., 2003). It supports the design of programs through modifiable object diagrams that are shared by a group of students who are collaboratively solving program design and programming assignments. Another example is that of Mendes et al. (2005) who have developed a collaborative learning environment for learning programming and data structures and algorithms.

Chapter 3

Overview on the Publications

During this research, I have studied the collaborative use of software visualizations. The research is based on empirical studies of students, who are learning programming or data structures and algorithms. This research has been reported in papers (**P1**, **P2**, **P3**, **P4**). These studies have utilized Jeliot 3, TRAKLA2, and BlueJ as the visualization tools during the learning and they have been performed at three different universities. In addition to this, I have developed better engagement capabilities into Jeliot 3 in the form of automatic question generation as reported in paper (**P5**) as well as integrated Jeliot 3 into a collaborative authoring tool Woven Stories to create a new tool called JeCo (Jeliot Collaboratively) as reported in paper (**P6**).

3.1 Summary of the Publications

In the first paper (**P1**), the research context for the whole thesis has been laid. Based on the analysis of previous literature, we propose to study collaborative learning with software visualizations, and in the scope of the paper, program visualization in particular. The paper contains three main results:

- a hypothesis how the engagement level between the learners and a visualization affects the collaborative learning process;
- an extension of the engagement taxonomy into the Extended Engagement Taxonomy (EET) to describe engagement in a finer level of granularity, and the use of the EET in the analysis of the collaborative learning process with visualizations;

- a partial empirical confirmation of the hypothesis in a study in which students were using program visualization tools, Jeliot 3 and BlueJ, during the collaborative learning of programming.

Based on previous results in empirical studies, the following hypothesis was formed: *increasing the level of engagement between learners and the visualization tool results in a higher positive impact of the visualization on the collaboration process.* The empirical study was carried out in the autumn of 2005 at the University of Joensuu in an introductory programming course. During the course students were solving programming exercises in small groups and using BlueJ and Jeliot 3 as the programming and program visualization tools. The students' learning processes were recorded using a video camera and a sample of the video materials was analyzed. To analyze the students' level of engagement with the visualization tools, the engagement taxonomy was extended in order to capture finer variations in the engagement.

A sample of the video material was classified according to four different categorizations: the engagement level of the visualization, students' activities, discussion contents, and transactive reasoning. The results showed that the engagement level and the students activities were correlated. For example, a higher level of engagement was positively correlated with more discussion and less silence. The distributions of the discussion contents were also qualitatively different depending on the level of engagement, i.e., when students were interacting with the visualization on a higher level of engagement their discussion contents were on a higher level of abstraction. Students seemed to concentrate more on lower level details, e.g., the program source code, when they were on lower levels of engagement. There were no statistically significant differences found regarding the transactive reasoning, however, higher levels of engagement had larger proportions of transactive reasoning. Furthermore, the reliability of the classifications in the discussion contents and transactive reasoning could not be guaranteed. Nevertheless, the results showed that the EET could be used to analyze the collaborative learning process and the levels of engagement were correlated positively with the collaborative activities of the students.

The second paper (**P2**) describes an empirical experiment that tested the hypothesis that the engagement level promoted by the algorithm visualization tool affects the learning results when students are collaboratively learning the concepts related to the binary heap. The experiment was carried out in a data structures and algorithms course at the University of Turku. The design of the experiment was a between-subjects design with one factor—the engagement level of the visualization—and the learning results were measured with pre- and post-tests. Between the tests students were learning the concepts of binary heap in pairs. The treatment group was using web-based material that consisted of text, static figures and visual algorithm simulation exercises from TRAKLA2 on the EET level *changing*. They could also view the model answers to the exercises. The control group had the same web-based material, but the visual algorithm simulation exercises were replaced by the model answer visualizations of the same VAS exercises. These visualizations were on the EET level *controlled viewing*. In addition to this, both groups had to answer an exercise sheet in which the exercises were related to the different aspects of the binary heap which the students were supposed to learn. The time for learning was controlled and it was expected that most of the students would be able to study the learning materials during the given time. The results of the experiment showed that although there was a difference in the learning outcomes of the students in favor of the treatment group, it was not statistically significant. Further analysis showed that students without any prior knowledge about the binary heap benefited more from the treatment based on the effect size (Cohen, 1977). Some methodological issues related to the pre- and post-tests were also determined in the analysis. In particular, the post-test should be made more challenging because the students had learned the basic topics well during the collaborative learning session.

The third paper (**P3**) reports on a replication of the study reported in (**P2**) at another university, Helsinki University of Technology. Compared to the previous study, the methodology was enhanced by changing the learning materials and the pre- and post-tests. Furthermore, the study reports on a post-hoc video analysis that revealed that, contrary to our instructions, some pairs in the treatment group never used the algorithm simulation exercises, but merely watched the model answer animations. On the one hand, this can be considered as a flaw of our study that we should have been able to control. On the other hand, this video analysis allowed us to detect this behavior and reclassify the students according to their real behavior changing the study setting from experimental to observational, as we partially lost the randomization of the students into the groups. Nevertheless, based on the post-hoc analysis of the background variables (pre-test score, programming course grades, and CS and Math credits studied so far) the groups were still equal on their previous knowledge and abilities related to the topic. Actually, they became even more equal, because after the randomization there was a significant difference in the programming course grades in favor of the treatment group, but this differences disappeared after the re-grouping. Furthermore, this reorganization of the students into treatment and control groups allowed us to obtain a statistically significant difference between the learning results of the new control and treatment groups.

The paper also contains information on how frequently the different EET levels were used during the learning process. It was found that the treatment group used visualizations more than the control group (as was expected) and the use of visualizations was on higher levels of engagement.

The fourth paper (**P4**) applies the methodology introduced in (**P1**) to analyze the video and audio material collected during the experiment reported in (**P3**). The rationale for this analysis was that this would allow us to explain the differences in the learning results between the groups and to demonstrate that the findings reported in (**P1**) are independent of the tool. In the video analysis, a sample of the screen and audio recordings from the period of time during which students were collaboratively learning about the heap data structure were analyzed. The level of engagement, the current activity and the discussion contents were recorded for each episode. The results indicated that the amount of discussion increased when the level of engagement increased. Furthermore, the discussion contents were more balanced across different topics on the higher levels of engagement. This is in line with the research reported in (**P1**) and also gives an explanation for the results in (**P3**), i.e., when students are learning with visualizations on higher level of engagement, they interact more with each other and more interaction has been shown to enhance learning results (Teasley, 1997).

In the fifth paper (**P5**), the rationale and development of an automatic prediction-type question generation to Jeliot 3 is discussed. The development of this feature in Jeliot 3 is based on the engagement taxonomy and previous results that show that the level *responding* enhances learning results compared to passive levels of engagement. Working at the *responding* level should facilitate the collaborative learning, because it prompts for conversation about the answers to each question and thus promotes meaningful conversations related to the topic that is being learned. Furthermore, it provides meaningful pauses to the visualization during which students can discuss what has happened in the visualization. From the implementation point of view, the architecture of Jeliot 3 and how it enabled the automatic generation of the prediction-type questions was discussed: the MCode program-trace description language could be preprocessed in order to produce prediction-type questions automatically. The work also built on the work of Rößling and Häussge (2004), using the question display and recording library they had developed. The paper also explains that the automatic questions could be used for automatic assessment and user modeling.

The sixth paper (**P6**)[1] introduced a new tool called *Jeliot Collaboratively (JeCo)*, which combines Jeliot 3 with *Woven Stories*, a collaborative story authoring environment, in order to allow users to collaboratively create programs and visualize them on-line. The motivation to create a new tool was to support distributed and collaborative programming, as well as learning to program in distance education. As it has been shown in previous research, pair programming and collaborative learning

[1]There is a mistake on the last page of the article: the first biography is Jussi Nuutinen's and the second one Niko Myller's, unlike printed.

can help students in the learning of programming and we expect that these effects are even stronger on distance education. However, because it is hard to maintain a common context for discussions in on-line collaboration, the visualizations can aid in the creation of a suitable context for collaboration and collaborative learning (Suthers et al., 2007b, 2008, 2003b). This environment can also be used to study the distributed collaborative learning of programming with visualizations by collecting the log data of the tool's usage (Suthers et al., 2007a, 2003b).

The reasons to select Jeliot 3 and Woven Stories to be integrated were discussed. Firstly, Jeliot 3 and it predecessors have been found educationally effective especially in introductory programming. Secondly, the novices need simpler tools to collaborate than integrated development environments for programming or version control and configuration management systems. Thirdly, both of these tools are open-source, which enables the integration of the systems in a meaningful way. The paper introduces the integrated tools and the means to combine. The features of the resulting software, JeCo, are explained and several future development areas are identified.

3.2 Contributions of the Author

The use of several visualization tools and studying their effects on collaborative learning in multiple contexts required that the work reported in this thesis be done in several institutions and in collaboration with several research groups and researchers. The research leading to the publication of (**P1**) was carried out at the University of Joensuu. I was the main author of the article and conducted the empirical study and its analysis. I received help for the practical arrangements of the study from Andrés Moreno. Roman Bednarik helped in conducting the study and was the second rater in order to test the inter-rater reliabilities. All the authors contributed to the study design and the interpretations of the results as well as to the writing process. Part of the video analysis was carried out while I was a visiting lecturer at Tumaini University, in Iringa, Tanzania.

The publications (**P2**, **P3**, **P4**) were a joint effort between me, Mikko-Jussi Laakso from the University of Turku and Ari Korhonen from the Helsinki University of Technology. The original idea for this series of studies was mine, and I proposed most of the methodology used in these studies. However, all authors contributed to the final design of the experiments. The experiments and data gathering were carried out together at the University of Turku (**P2**) and at the Helsinki University of Technology (**P3**, **P4**) and the analysis of the results and reporting was done in collaboration.

In paper (**P5**), the development and writing was done by me. The development work was partially done while I was a visiting researcher at Massey University in Palmerston North, New Zealand, hosted by Professor Kinshuk.

The development work reported in (**P6**) was done by me, but I received valuable help from Jussi Nuutinen, who also contributed to writing the paper, especially in parts that were related to the *Woven Stories*.

Chapter 4

Summary of the Results

"The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' but 'That's funny...'"
– Isaac Asimov

In this chapter, the main results of the original publications (**P1**)–(**P6**) are summarized and compared with the results obtained in literature. The results of this thesis are three-fold: theoretical and methodological, empirical and practical. The extension of the engagement taxonomy and its hypotheses as well as its use in analysis of collaborative learning with visualizations is discussed in Section 4.1. Section 4.2 explains the empirical studies of collaborative learning with visualizations. The applications of the theory and the results of the empirical studies are described in Section 4.3.

4.1 Extended Engagement Taxonomy

Our motivation to extend the engagement taxonomy and the hypotheses attached to it was to be able to describe the finer variations of the engagement when students are collaboratively learning with software visualization tools. In addition to this, we wanted to study how the engagement levels promoted by the visualization tool might affect the collaboration process and its outcomes. This led to the development of extended engagement taxonomy (EET) that is shown in Table 4.1 and the hypotheses which state that an increase in the engagement level between the collaborating learners and the visualization tool results in better collaboration process with more discussion and sharing (**P1**, **P4**), and improves the learning results of the students who are learning collaboratively with visualizations (**P2**, **P3**).

From a methodological point of view, EET was used to classify episodes in a collaborative learning process based on the engagement level promoted by the SV tool during each episode (**P1**, **P3**, **P4**). This was done to analyze the relationship between the activities and the discussion contents of the students on varying levels of engagement and to find support for the hypotheses. This can be seen as a methodological contribution of this research.

The engagement taxonomy was introduced by Naps et al. (2002) and since then, research has been carried out to test the hypotheses it provided (Urquiza-Fuentes and Velázquez-Iturbide, 2007, Lauer, 2006, Rhodes et al., 2006, Grissom et al., 2003, Naps and Grissom, 2002). By analyzing the new research results, Lauer (2008b,a) has also proposed modifications and addition to the engagement taxonomy, which are similar to the modifications and additions proposed in the EET. These modification are summarized and compared to the EET below.

- Splitting the level of *viewing* into two: *active viewing* and *passive viewing* (Lauer, 2008a). These levels are similar to *viewing* and *controlled viewing* in the EET, respectively. This is also recognized by Lauer (2008b).

- Splitting the level of *constructing* into *simulating*, *hand-constructing* and *code-based constructing.* The levels of *hand-constructing* and *code-based constructing* are comparable to the levels of *constructing* and *modifying* in the EET, respectively. The level of *simulating* is missing from EET.

- The proposal of Lauer (2008b) is missing comparable levels for *entering input* and *reviewing.*

The work by Lauer (2008b,a) gives an empirical validation to some of the proposed extensions in the EET. I was unaware of the research of Lauer (2008b,a), and *vice versa*, until we met in the poster presentation of Lauer (2008a). Therefore, the research of the both of us should be regarded as complimentary and verifying the work that we have done independently.

In retrospect, the engagement level of *simulating* as proposed by Lauer (2008b) would have been more suitable for the visual algorithm simulation exercises of TRAKLA2 compared to the level *changing* that was used in the papers. The level *simulating* should be used when reporting future research; however, in this thesis the EET level *changing* is used in order to maintain clarity and consistency.

Table 4.1: The extended engagement taxonomy. The levels marked with (*) are from the original engagement taxonomy although definitions of the *viewing* and *changing* levels have been modified (**P1**).

Engagement level	Description
No viewing (*)	There is no visualization to be viewed, but only material in textual format. For example, the students are reviewing the source code without modifying it or they are looking at the learning materials.
Viewing (*)	The visualization is viewed with no interaction. For example, the students are looking at the visualization or the program output or a static image.
Controlled viewing	The visualization is viewed and the students control the visualization, for example by selecting objects to inspect or by changing the speed of the animation. This has been deemed important, for instance by Rößling and Naps (2002).
Entering input	The user enters input to a program or parameters to a method before or during their execution.
Responding (*)	The visualization is accompanied by questions that are related to the contents of the visualization.
Changing (*)	The visualization and its components are provided, but the changing of the visualization is allowed, for instance, by direct manipulation.
Modifying	There is a possibility to change the visualization before it is viewed, for example, by changing the source code of a program to be visualized or a data structure that is being used.
Constructing (*)	The visualization is created by the student from components such as text and geometric shapes.
Presenting (*)	Visualizations are presented and explained to others for feedback and discussion.
Reviewing	Visualizations are viewed for the purpose of providing comments, suggestions and feedback on the visualization itself or on the program or algorithm.

4.2 Analysis of Collaborative Learning with Software Visualizations

The results related to the analysis of collaborative learning with software visualizations can be divided into two parts. The findings about the collaborative learning process with SV tools are discussed in Section 4.2.1. In Section 4.2.2, the effects of collaborative learning with SV tools on learning outcomes are reviewed and further analyzed.

4.2.1 Collaborative Learning Process with Software Visualizations

The theoretical framework related to the study of the collaborative learning process with software visualization has been explained in (**P1**, **P4**). It presents the hypothesis that the engagement level of a visualization has an effect on the collaborative learning process, i.e., the higher the engagement, the more interaction there is between the learners. This hypothesis has been tested with empirical studies reported in (**P1**, **P4**). These studies have shown that the level of engagement promoted by the visualization and the students' activities are correlated and that this holds with three different SV tools. This can be seen when comparing the results from the studies reported in (**P1**, **P4**) and summarized in Figure 4.1. The only data point that does not fit into this model is the EET level *no viewing* when students are collaborative learning with Jeliot 3. This can be at least partially explained by the fact that the animations in Jeliot 3 are so intensive that students might not have free cognitive resources in order to discuss the issues related to the animations because they need to concentrate on following the animation, i.e., they were overly engaged. Therefore, the discussion that was related to the animations (i.e., concerning the levels *viewing* and *entering input*) often happened after the visualization had finished, i.e., when the EET level was *no viewing.* However, this discrepancy should be further analyzed in future studies. One way to solve this issue would be to stop the animation in Jeliot 3 in key locations and ask a question related to the animation, as will be discussed in Section 4.3.

Tables 4.2 and 4.3 summarize how the different engagement levels are used during the learning process. This offers another view in order to understand how the visualization tools are used in collaboration and how they affected it. Based on the results in Table 4.2, the primary mode of using Jeliot is *viewing* the visualization it provides and when added to *entering input* level (which also takes place during the playing of the animation), the animation is active over 85% of the time that Jeliot 3 is used. In contrast, the distribution of the time on different EET levels is more balanced between *no viewing*, *viewing* and *entering input* when BlueJ is used. Furthermore, BlueJ supports higher levels of engagement than Jeliot 3, although they were used quite rarely by the students. The level *no viewing* in both of these tools refers either to the situation where the students are looking at the source code or to the situation where they are viewing learning materials during the use of the tool.

The use of the TRAKLA2 algorithm animations (i.e., model answers) on the EET

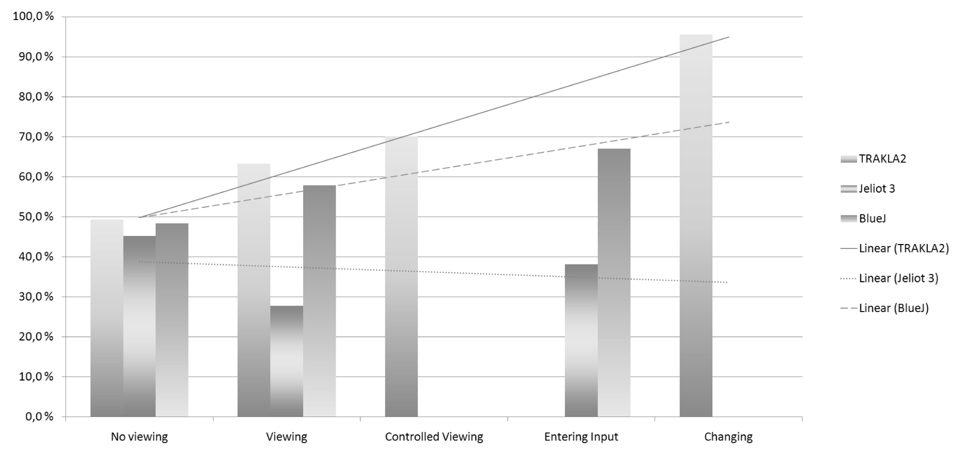


Figure 4.1: The proportions of discussions from all activities on the different EET levels when learning with different visualization tools from (Myller et al., 2009, Korhonen et al., 2009).

Table 4.2: The distribution of time between EET levels when using either Jeliot 3 or BlueJ as reported in (**P1**).

	No view.	**View.**	**Cont. view.**	**Enter. input**	**Chang.**	**Mod.**	**Const.**
Jeliot 3	11.8%	69.5%	3.4%	15.4%	N/A	N/A	N/A
BlueJ	33.1%	26.2%	6.3%	29.8%	0.6%	1.7%	2.5%

level *controlled viewing* and the exercises on the EET level *changing* are described in Table 4.3. The EET levels *no viewing* and *viewing* refer to the learning materials and to the static images in the learning materials, respectively. The Table shows that the treatment group used the visualizations for a longer time compared to the control group. The treatment group was using 12% of their time to solve the visual algorithm simulation exercises, which were not available to the control group. In addition to this, the treatment group used the model answer visualizations for 5% of their time and the control group for 13%. For the control group these were the only dynamic visualizations available. The static visualizations were used for the same amount of time and the textual information was used less by the treatment group.

The discussion contents during the collaboration were qualitatively described in (**P1**, **P4**). The key findings were that the distribution of the discussion topics seemed to be different between the levels of engagement. Firstly, the abstraction level of the discussions seemed to vary; for instance, when students were learning with the lowest engagement level, the discussion was about the low level details of a

Table 4.3: The distribution of time between EET levels when using different versions of TRAKLA2 environment as reported in (**P3**).

	No viewing	**Viewing**	**Controlled viewing**	**Changing**
TRAKLA2 Animations	48.1%	38.1%	13.8%	N/A
TRAKLA2 Exercises	43.2%	38.3%	5.9%	12.6%

program or algorithm and when students were on the higher levels of engagement, they discussed more often topics that are on higher level of abstraction. In between, the students discussed topics in a more balanced way. In (**P1**), the amount of transactive reasoning was also analyzed and no statistically significant differences were found, although the higher levels of engagement contained more transactive reasoning. However, we were unable to achieve high enough inter-rater reliabilities for the discussion contents and transactive reasoning categories in the research reported in (**P1**), and thus, the results are questionable and should be further evaluated in future research.

Lavonen et al. (2002) have studied elementary school students' behavior during collaborative programming and problem solving when they were using the visual programming environment Empirica Control. The methodology and findings from that study are in many ways similar to the studies reported in this thesis (**P1**, **P4**). The methodology used to classify the episodes on the video materials was similar in all studies. Lavonen et al. (2002) categorized the task type of each episode as well as the type and nature of the interaction during the episode. Although they did not use the engagement level, but rather the task type, as a measure to classify the episodes, those task types can be at least partially reclassified to the different EET levels retrospectively. There were six main task types: problem identification, formulation and specification, recognizing and finding facts and resources, planning, generating and evaluating ideas, constructing, testing and debugging (i.e., evaluating). When analyzing those tasks from the perspective of the EET, the visual construction of a program can be positioned on the EET level *constructing*, the planning is at least partially on the levels of *modifying* and *viewing*, and the testing and debugging is probably alternating between the EET levels *viewing*, *entering input* and *modifying.* The other task types, i.e., generating and evaluating ideas, problem identification, formulation and specification, and recognizing and finding facts and resources, are operating on the EET levels *no viewing* and *viewing.* The results of the study show that the largest proportions of the reciprocal interactions between students were observed when the students were generating and evaluating ideas, testing and debugging, planning, and constructing the program visually using icons, in that order. Furthermore, the overall count of the interactions in the generating and evaluating ideas was small (2.2%) compared to the other task types with the highest amount of interaction. Therefore, it can be said that those task types that were operating on higher levels of engagement contained more mutual interaction between the students, which is in line with the findings reported in (**P1**, **P4**).

The results reported in this section as well as in the papers (**P1**, **P4**) are comparable to the results of Hundhausen and Brown (2008), who showed that those students who were using an algorithm visualization tool on higher engagement levels were spending less time unproductively and concentrating more on the solution (cf. silent vs. discussing). Furthermore, those students needed less help from their teacher. In the paper (**P1**), it was shown that the students were discussing with the instructor most often when the engagement was on the level of *no viewing*.

4.2.2 Learning Outcomes of the Collaborative Learning with Software Visualizations

The learning results of the students who have been learning collaboratively with visualizations are reported in (**P2**, **P3**). One of the hypotheses related to the EET is that the students who collaboratively learn with visualizations on a higher engagement level achieve better learning outcomes compared to the students who learn with visualizations on a lower engagement level.

The experiment reported in (**P2**) analyzed this hypothesis. The pre-test results and post-hoc background variable analysis showed that the treatment and the control group were equal. The post-test results from the experiment reported in (**P2**) are shown in Table 4.4. The results show that the treatment group was consistently better in the questions of the post-test. However, based on the analysis of the post-test results and the learning gains, i.e., the average of the differences between the students' results of the pre- and post-tests, there were no statistically significant differences found in the learning results. To further analyze the benefits of the visualizations in the collaborative learning, students were divided into two groups based on their pre-test results: students with previous knowledge and students without previous knowledge of the concepts related to the binary heaps. The subsequent analysis showed that, although the differences were not statistically significant, the effect size (Cohen, 1977) of the group without previous knowledge was medium indicating that a statistically significant difference could be found with the larger sample. Furthermore, there were issues related to the learning materials and questions used in the pre- and post-tests that could have affected the results; therefore, the study was replicated and the replication was reported in (**P3**).

In the study presented in (**P3**), the collaborative learning outcomes were also analyzed by using pre- and post-tests. It was originally designed as an experiment,

Table 4.4: The post-test results from the experiment reported in (**P2**). Note that the results on questions 7 and 8 were not reported in (**P2**), because the comparison was limited to the questions that were the same in pre- and post-test.

	Control	**Treatment**
Question 1	2.6 (1.9)	2.8 (1.8)
Question 2	3.2 (1.6)	3.3 (1.4)
Question 3	3.8 (0.9)	4.0 (0.0)
Question 4	2.3 (1.3)	2.8 (0.9)
Question 5	2.5 (1.6)	2.9 (1.4)
Question 6	3.6 (1.1)	3.7 (1.0)
Question 7	3.5 (1.2)	3.3 (1.3)
Question 8	3.1 (1.5)	2.1 (1.9)
Total	24.5 (6.6)	24.9 (5.2)

but in a post-hoc video analysis it was found that some students were not behaving as expected, i.e., some student pairs were not using the visual algorithm simulation exercises at all, but only viewed the corresponding model answer visualization. Therefore, we needed to classify the students according to their behavior, making the study observational rather than an experiment. The results from the groups determined based on the observations are shown in Table 4.5. The treatment group was consistently better than the control group in 11 questions of the post-test. Furthermore, it was found that the differences in the total and the pair-averaged total were statistically significant using the one-tailed t-test. This supports the hypothesis that higher engagement level positively affect the learning results.

In retrospect, the overall results from both experiments can also be analyzed with a binomial test similar to the analyses proposed by Bednarik and Randolph (2008). However, we need to make a few assumptions: each of the questions is equally important and independent of the other questions, and the questions from the two separate experiments are comparable and aggregable.

Firstly, we count the number of questions in which the treatment group is better than, or at least equal to, the control group from all the questions in the post-tests. For the experiment reported in (**P2**), this proportion is 6 questions out of 8 questions in the post-test, and for the study reported in (**P3**), it is 10 questions out of 13 questions in the post-test. When these statistics are combined, we get that in 16 questions out of 21 questions in the post-tests the treatment group outperformed the control group.

$$F(x, n, p) = \Pr(X \geq x) = \sum_{i=x}^{n} \binom{n}{i} p^i (1-p)^{n-i} \tag{4.1}$$

Table 4.5: Post-test results from the observational study reported in (**P3**).

	Control	**Treatment**
Question 1	2.46 (1.61)	2.33 (1.80)
Question 2	1.78 (1.23)	2.19 (1.29)
Question 3	3.76 (0.89)	4.00 (0.00)
Question 4	2.32 (1.42)	2.33 (1.59)
Question 5	2.62 (1.47)	3.38 (0.92)
Question 6	3.82 (0.80)	4.00 (0.00)
Subtotal	16.76 (4.49)	18.24 (3.56)
Question 7	3.96 (0.20)	3.43 (1.29)
Question 8	3.44 (1.13)	3.76 (0.89)
Question 9	2.36 (0.85)	2.67 (0.91)
Question 10	2.20 (1.29)	2.62 (1.40)
Question 11	0.54 (1.31)	1.10 (1.70)
Question 12	0.92 (1.64)	1.24 (1.84)
Question 13	0.28 (0.67)	0.29 (0.46)
Total	30.46 (6.54)	33.33 (6.71)
Pair Averaged Total	30.42 (4.55)	33.45 (4.34)

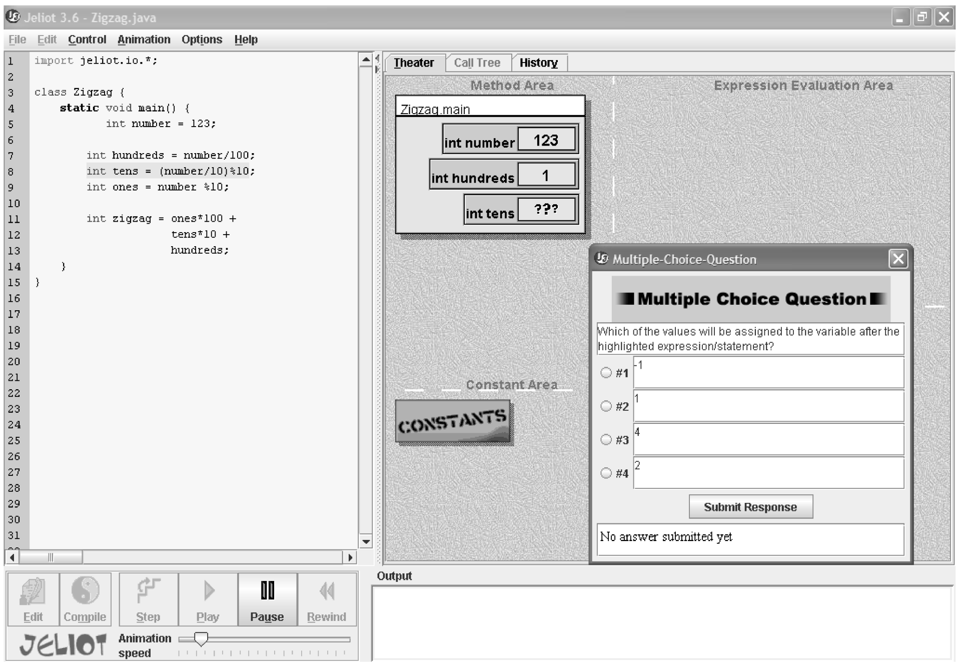
The null hypothesis for the binomial test is that the treatment did not produce any learning effects, i.e., it is equally likely for either group to perform better in the post-test. We can now compute the p-value for getting higher points than the other group in 16 questions when there are a total of 21 questions and the probability to get higher points in the question is equal between the control and treatment groups (i.e., 50% probability). This can be calculated by using the formula in Equation 4.1. The result of the binomial test ($F(16, 21, 0.5) = 0.013$, two-tailed $p < 0.05$) indicates that we can reject the null hypothesis and claim that the treatments had a statistically significant positive effect on the learning results of the students compared to the control group when measured by the questions in the post-tests.

This gives further support to the hypothesis that the level of engagement promoted by a visualization has a positive effect on the learning results when student are learning collaboratively with visualizations. The effect is that the learning outcomes are better when the visualization is used on higher level of engagement during the learning process compared to the control group.

These findings are in line with the results of Hundhausen and Brown (2008) who showed that when student pairs were working on implementing an algorithm using either a text editor working on EET level *no viewing* or an algorithm visualization tool ALVIS working on EET levels *viewing* and *constructing*, the latter group created better code than the former. The results reported in this section also are also in line with the original research framework of ET developed by Naps et al. (2002).

4.3 Applications

The results of the empirical study (**P1**) pointed out that the program animations in Jeliot 3 should be enhanced with techniques that would raise the engagement level of the visualization and allow meaningful pauses in the animation in order to support discussion. As an attempt to solve this issue, (**P5**) described how automatic prediction-type question generation has been added to Jeliot 3 and how it could be used. Figure 4.2 shows an example of a multiple-choice question, which has been generated automatically.



```
import jeliot.io.*;

class Zigzag {
    static void main() {
            int number = 123;

        int hundreds = number/100;
        int tens = (number/10)%10;
        int ones = number %10;

        int zigzag = ones*100 +
                     tens*10 +
                     hundreds;
    }
}
```

Which of the values will be assigned to the variable after the highlighted expression/statement?

#1 -1

#2 1

#3 4

#4 2

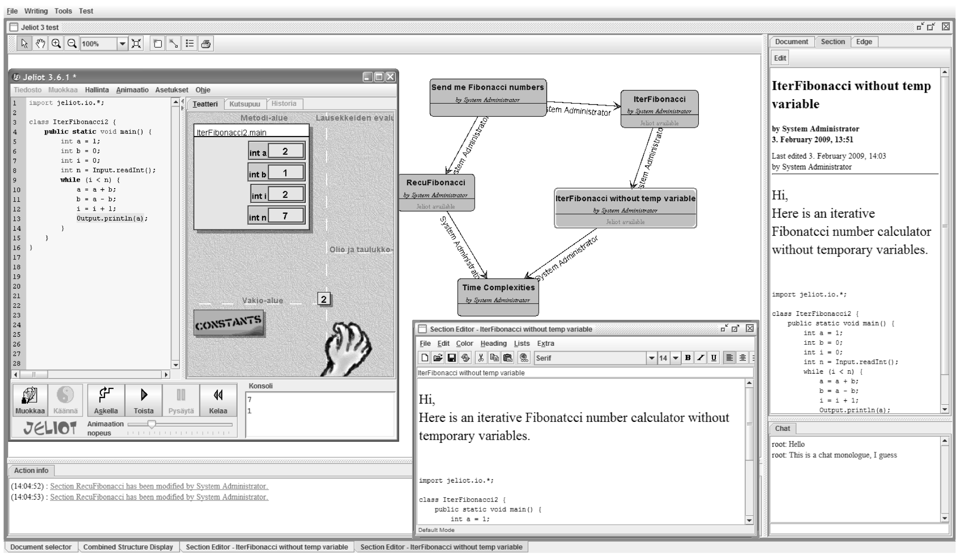
Submit Response

No answer submitted yet

Figure 4.2: The automatically generated prediction-type questions are raised during the program visualization.

The automatic question generation can be used for multiple purposes. For instance, it can support the individual and collaborative learning with visualizations or it can be used as a quiz or test for students after they have learned a certain concept. Furthermore, it can be connected to a user-model facility as described by Moreno et al. (2007a) and update the user model based on the answers of the student, in order to allow adaption of the visualization. Currently, the educational effectiveness of the automatic prediction-type question generation in Jeliot 3 has not been studied and this should be performed in order to understand how question generation affects learning.

In order to support the collaborative use of Jeliot 3 in distance education, Jeliot 3 was combined with a collaborative authoring tool called Woven Stories, which supports both synchronous and asynchronous modes of communication and sharing in on-line collaborative authoring. (**P6**) describes JeCo (Jeliot Collaboratively), a tool integrating Jeliot 3 and Woven Stories, and how it can enhance on-line collaborative learning in programming. Figure 4.3 illustrates the user interface of JeCo. One can add a new node (i.e., text or program code) to the graph of related messages or sections, and connect it to the previous nodes by using edges. If the node contains program source code, a sign "Jeliot available" is shown below the title of the node. By right-clicking the node, the user can select to copy the source code into Jeliot and play the animation of the program execution in Jeliot 3. Furthermore, the contents of the node are shown on the right-hand side in a content frame and below it there is a chat window. The nodes can be edited by the owner of the node and the contents can consist of rich text and program source code.



```
import jeliot.io.*;

class IterFibonacci2 {
    public static void main() {
        int a = 1;
        int b = 0;
        int i = 0;
        int n = Input.readInt();
        while (i < n) {
            a = a + b;
            b = a - b;
            i = i + 1;
            Output.println(a);
        }
    }
}
```

```
7
1
```

IterFibonacci without temp variable

by System Administrator
3. February 2009, 13:51

Last edited 3. February 2009, 14:03
by System Administrator

Hi,
Here is an iterative Fibonatcci number calculator without temporary variables.

root: Hello
root: This is a chat monologue, I guess

(14:04:52) : Section RecuFibonacci has been modified by System Administrator.
(14:04:53) : Section RecuFibonacci has been modified by System Administrator.

The visualization of the discussions and collaboration shows the different messages and their relationships with each other as a graph. The visualization is similar to the graph representation used by Suthers et al. (2007b, 2008) in the context of on-line collaborative inquiry. They have shown that the graph representation, which supports both task completion as well as communication, is better for focusing the discussions on the task-related issues. This also affected the learning results when compared to the threaded discussions, which are accompanied with a graph representation that was only supposed to be used to complete the task. In JeCo, the representation graph is mainly a communications channel, which shows the structure of past collaborations and messages, and their relations. JeCo also provides a chat facility which can be used during synchronous collaborative learning. This tool can be used to support collaborative learning of programming with visualization in distance education. Furthermore, it provides an interesting research tool in this context as it saves all the interactions into a log file and database so that this data is available to a researcher.

Chapter 5

Discussion

"What is a scientist after all? It is a curious man looking through a keyhole, the keyhole of nature, trying to know what's going on."
– Jacques Yves Cousteau

In this thesis, various aspects of the collaborative learning of programming and algorithms and data structures with software visualizations have been studied. The key findings are the extension of the engagement taxonomy and its utilization in the analysis of collaborative learning with software visualization. It was found that the engagement promoted by the visualization has a positive effect on the collaborative learning process of students as well as its learning outcomes, and the engagement seems to be related even to the discussion contents, but this should be further analyzed in the future research. Based on these results, extensions to the existing program visualization tool Jeliot 3 were designed and implemented. The implications of the studies and results to the teaching and research practices are discussed in Section 5.1. The concluding remarks are given in Section 5.2 and the introduction and summary part of this thesis will end with a discussion on future perspectives in Section 5.3.

5.1 Implications of the Results

A general implication of the research is that the use of visualization in learning programming or data structures and algorithms is beneficial, when students learn with visualizations that support active forms of engagement. This use seems to have an effect not only on the learning outcomes but also on the collaborative learning process. This should inform the teaching of these subjects, in the sense that SV tools can and should be used during collaborative learning or pair programming.

Furthermore, SV tools that operate on higher levels of engagement should be favored, as well as the tools which support multiple levels of engagement.

SV tool developers can use these results since they emphasize the need for different kinds of interaction and engagement possibilities in visualizations in order to support collaboration. The extended engagement taxonomy provides new directions for possible engagements, and tools that support these various levels should be developed in order to evaluate the effects of those levels on learning.

There are two methodological issues that should be noted by researchers of SV tools in future studies. Firstly, the learning processes should be studied in combination of the learning outcomes. As can be seen from the results, there are several things that can be learned from this kind of analysis, for example, the description of the learning process gives meaning to the differences in the learning outcomes. Secondly, although the use of software for capturing computer screens and audio recording is not new, we have used these tools for another purpose: to monitor that students are behaving as they are expected to. In the research reported in (**P3**), it was found that the students who were supposed to learn with the visual algorithm simulation exercises never used them but only viewed the model answer animations. This was only detected during the post-hoc video analysis, and then video analysis was used to divide the students into groups based on their behavior. Therefore, it is recommended that every study analyzing the use or the educational effectiveness of SVs should use screen capturing in order to detect misbehavior or possible confounding factors. This also casts some doubt on previous results in the field, i.e., how we can be sure that the inconclusive results of the previous research were not partially due to the same issue?

There are some limitations to the validity of the results reported in this thesis. The analysis of the effects of engagements did not utilize all the engagement levels or all the different tools, so there is a possibility that the effects attributed to the engagement levels in general were due to those particular engagement levels or to the tools that were analyzed. However, I tried to eliminate these possibilities by utilizing multiple tools and by carrying out the studies in several contexts. The constructive research in the further development of Jeliot that was performed as part of the thesis has not been evaluated in the empirical studies. Although it was guided by the empirical studies that have been carried out during the research, the implementations have not yet been tested for educational effectivity. Therefore, the claims that those tools are educationally effective are based on the previous research and literature and not empirical data.

5.2 Concluding remarks

Research on the use of software visualization tools during collaborative learning is multi-disciplinary, and theories and methods from several fields of study need to be applied. In this work, I have applied methodology and theories from behavioral and educational sciences, as well as from computer science education research and research on computer-supported collaborative learning. From a practical point of view, I have studied students who have been learning introductory computer science. I have also used three different SV tools in the studies and performed them at three universities. This has given a rich context for this research that would not have been possible to achieve by using a single tool at one university.

Although there are claims that visualization tools are used too little outside the university where the tool is developed and that there are several forms of resistance to the adoption of these tools (Naps et al., 2002, Ben-Bassat Levy and Ben-Ari, 2007), this kind of research collaboration should make it easier to adopt new tools at multiple universities. Furthermore, research is need to inform the instructors of the possible benefits of the visualization tools, especially now, when we start to understand how and why the visualizations affect the learning outcomes.

From a point of view of the Jeliot 3, this research could be seen as action research designed to enhance the educational effects of Jeliot. Although Jeliot 3 was used in one of the studies, other tools BlueJ and TRAKLA2 were studied. The findings from research on the other SV tools can then be taken back to enhance and develop Jeliot 3 and this mutual feedback of research and development should continued.

5.3 Future Perspectives

The research reported in this thesis has also pointed to or resulted in several new research directions that could be pursued in the future.

As identified in Section 4.1, the extended engagement taxonomy (**P1**) and the additions proposed by Lauer (2008b) should be unified and further analyzed. Furthermore, it would be meaningful to make the taxonomy easily extensible and possibly even to divide it into multiple dimensions. Moreover, it should be analyzed how the communicative dimensions by Hundhausen (2005) are related to the engagement levels and what their role is in supporting communications and collaboration.

The methodology that was used in (**P1**) to analyze categories of transactive reasoning and discussion contents should be refined and further analyses of collaborative learning processes with visualizations should be carried out. This would enable us to analyze the correlation or causal relationships between the engagement levels and the transactive reasoning and discussion contents on various levels of engagement. It would also be interesting to analyze which aspects of the visualization might affect the discussion contents and what representational guidance is involved.

As discussed in (**P1**), it would be meaningful to study the long-term effects of the collaborative use of visualizations in programming and data structures and algorithms learning by carrying out a longitudinal study and comparing the amount of discussions and their contents over several weeks. Currently, there would even be available materials to carry out this kind of study.

The tools and techniques that were developed during this research should be evaluated. The educational effectiveness of the automatic prediction-type question generation should be tested in an empirical study in both individual and collaborative learning situations. Furthermore, the use of JeCo should be investigated and a methodology, similar to that used by Suthers et al. (2008), could be used to carry out the study and the analysis.

It would be interesting to develop and evaluate completely new interaction capabilities in Jeliot 3. For example, visual program simulation support could be added to Jeliot in order to promoted the EET level *changing* or *simulating*. This could be also an interesting combination of the research and development carried out by the two research groups who developed TRAKLA2 and Jeliot 3, because the former has knowledge of visual algorithm simulation and the latter of program visualization.

References

Baecker, R. (1981). Sorting out Sorting. Videotape, 30 minutes, presented at ACM SIGGRAPH '81 and excerpted in ACM SIGGRAPH Video Review #7.

Beck, L. L. and Chizhik, A. W. (2008). An experimental study of cooperative learning in CS1. In *SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education*, pages 205–209, New York, NY, USA. ACM.

Bednarik, R., Moreno, A., and Myller, N. (2006c). Various Utilizations of an Open-Source Program Visualization Tool, Jeliot 3. *Informatics in Education*, 5(2):267–276.

Bednarik, R., Myller, N., Sutinen, E., and Tukiainen, M. (2005a). Applying Eye-Movement Tracking to Program Visualization. In *Proceedings of 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*, pages 302–304, Dallas, Texas, USA. IEEE press.

Bednarik, R., Myller, N., Sutinen, E., and Tukiainen, M. (2005b). Effects of Experience on Gaze Behaviour during Program Animation. In *Proceedings of the 17th Annual Psychology of Programming Interest Group Workshop (PPIG'05)*, pages 49–61.

Bednarik, R., Myller, N., Sutinen, E., and Tukiainen, M. (2006a). Analyzing Individual Differences in Program Comprehension. *Technology, Instruction, Cognition and Learning*, 3(3):205–232.

Bednarik, R., Myller, N., Sutinen, E., and Tukiainen, M. (2006b). Program Visualisation: Comparing Eye-tracking Patterns with Comprehension Summaries and Performance. In *Proceedings of the 18th Annual Workshop of the Psychology of Programming Interest Group (PPIG'06)*, pages 68–82.

Bednarik, R. and Randolph, J. (2008). Studying Cognitive Processes in Program Comprehension: Levels of Analysis of Sparse Eye-tracking Data. In Hammoud, R. I., editor, *Passive Eye Monitoring: Algorithms, Applications and Experiments*, pages 372–386. Springer.

Ben-Ari, M., Myller, N., Sutinen, E., and Tarhio, J. (2002). Perspectives on Program Animation with Jeliot. In Diehl, S., editor, *Software Visualization, Lecture Notes in Computer Science, Volume 2269*, pages 31–45, Berlin. Springer-Verlag. (SpringerLink publication).

Ben-Bassat Levy, R. and Ben-Ari, M. (2007). We work so hard and they don't use it: Acceptance of software tools by teachers. *SIGCSE Bulletin*, 39(3):246–250.

Ben-Bassat Levy, R., Ben-Ari, M., and Uronen, P. A. (2003). The Jeliot 2000 program animation system. *Computers & Education*, 40(1):15–21.

Bennedsen, J. and Caspersen, M. E. (2007). Failure rates in introductory programming. *SIGCSE Bulletin*, 39(2):32–36.

Berkowitz, M. W. and Gibbs, J. C. (1983). Measuring the development of features in moral discussion. *Merill-Palmer Quarterly*, 29:399–410.

Boroni, C. M., Eneboe, T. J., Goosey, F. W., Ross, J. A., and Ross, R. J. (1996). Dancing with DynaLab: Endearing the Science of Computing to Students. *SIGCSE Bulletin*, 28(1):135–139.

Bryant, S., Romero, P., and du Boulay, B. (2005). Pair programming and the re-appropriation of individual tools for collaborative programming. In Pendergast, M., Schmidt, K., Mark, G., and Ackerman, M., editors, *Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work, GROUP 2005*, pages 332–333. ACM Press.

Byrne, M., Catrambone, R., and Stasko, J. (1999). Evaluating animations as student aids in learning computer algorithms. *Computers & Education*, 33(4):253–278.

Chi, M. T. H. (1997). Quatifying qualitative analyses of verbal data: A practical guide. *The Journal of The Learning Sciences*, 6:271–315.

Chinn, D., Martin, K., and Spencer, C. (2007). Treisman workshops and student performance in cs. In *Proceedings of the 28th SIGCSE Technical Symposium on Computer Science Education*, pages 203–207.

Cohen, J. (1977). *Statistical power analysis for the behavioral sciences.* Academic Press, New York.

Crescenzi, P., Demetrescu, C., Finocchi, I., and Petreschi, R. (2000). Reversible execution and visualization of programs with LEONARDO. *Journal of Visual Languages and Computing*, 11(2):125–150.

Dillenbourg, P. and Self, J. (1992). A computational approach to socially distributed cognition. *European Journal of Psychology of Education*, 7(4):352–373.

Ebel, G. and Ben-Ari, M. (2006). Affective effects of program visualization. In *Second International Computing Education Research Workshop*, pages 1–5, Canterbury, UK. ACM Press.

Evans, C. and Gibbons, N. J. (2007). The interactivity effect in multimedia learning. *Computers & Education*, 49(4):1147–1160.

Fincher, S. and Petre, M. (2004). *Computer Science Education Research.* Routledge.

Fisker, K., McCall, D., Kölling, M., and Quig, B. (2008). Group work support for the bluej ide. In *ITiCSE '08: Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pages 163–168, New York, NY, USA. ACM.

Gall, M. D., Gall, J. P., and Borg, W. R. (2006). *Educational Research: An Introduction*. Allyn & Bacon, 8th edition.

Green, T. R. G. and Petre, M. (1996). Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *Journal of Visual Languages and Computing*, 7:131–174.

Grissom, S., McNally, M., and Naps, T. L. (2003). Algorithm visualization in CS education: comparing levels of student engagement. In *Proceedings of the First ACM Symposium on Software Visualization*, pages 87–94.

Grudin, J. (1994). Computer-supported cooperative work: history and focus. *Computer*, 27(5):19–26.

Haaster, K. V. and Hagan, D. (2004). Teaching and learning with BlueJ: An evaluation of a pedagogical tool. In *Proceedings of Informing Science + IT Education Conference (InSITE 2004)*, pages 455–470, California, USA. Informing Science Institute.

Hillion, S. (2002). DynamicJava. WWW-page. `http://sourceforge.net/projects/djava/` (Accessed 2009-04-01).

Hübscher-Younger, T. and Narayanan, N. H. (2003). Constructive and collaborative learning of algorithms. *SIGCSE Bulletin*, 35(1):6–10.

Hundhausen, C. D. (2002). Integrating Algorithm Visualization Technology into an Undergraduate Algorithms Course: Ethnographic Studies of a Social Constructivist Approach. *Computers & Education*, 39(3):237–260.

Hundhausen, C. D. (2005). Using end-user visualization environments to mediate conversations: A 'Communicative Dimensions' framework. *Journal of Visual Languages and Computing*, 16(3):153–185.

Hundhausen, C. D. and Brown, J. L. (2007a). What You See Is What You Code: A 'live' algorithm development and visualization environment for novice learners. *Journal of Visual Languages and Computing*, 18(1):22–47.

Hundhausen, C. D. and Brown, J. L. (2008). Designing, visualizing, and discussing algorithms within a CS 1 studio experience: An empirical study. *Computers & Education*, 50(1):301–326.

Hundhausen, C. D., Douglas, S. A., and Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing*, 13(3):259–290.

Jadud, M. C. (2006). Methods and tools for exploring novice compilation behaviour. In *ICER '06: Proceedings of the second international workshop on Computing education research*, pages 73–84, New York, NY, USA. ACM.

Jain, J., II, J. H. C., Hendrix, T. D., and Barowski, L. A. (2006). Experimental evaluation of animated-verifying object viewers for java. In *SoftVis '06: Proceedings of the 2006 ACM symposium on Software visualization*, pages 27–36, New York, NY, USA. ACM.

Janssen, J., Erkens, G., Kanselaar, G., and Jaspers, J. (2007). Visualization of participation: Does it contribute to successful computer-supported collaborative learning? *Computers & Educucation*, 49(4):1037–1065.

Jarc, D., Feldman, M., and Heller, R. (2000). Assessing the benefits of interactive prediction using web-based algorithm animation courseware. In *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education*, pages 377–381, Austin, Texas, USA.

Järvinen, P. (2004a). *On research methods.* Opinpajan kirja, Tampere.

Järvinen, P. (2004b). Research questions guiding selection of an appropriate research method. Technical Report D-2004-5, Department of Computer Science, University of Tampere.

Jehng, J.-C. J. and Chan, T.-W. (1998). Designing computer support for collaborative visual learning in the domain of computer programming. *Computers in Human Behavior*, 14(3):429–448.

Johnson, R. B. and Onwuegbuzie, A. J. (2004). Mixed methods research: A research paradigm whose time has come. *Educational Researcher*, 33(7):14–26.

Jordan, B. and Henderson, A. (1995). Interaction analysis: Foundations and practice. *The Journal of the Learning Sciences*, 4(1):39–103.

Kannusmäki, O., Moreno, A., Myller, N., and Sutinen, E. (2004). What a Novice Wants: Students Using Program Visualization in Distance Programming Course. In Korhonen, A., editor, *Proceedings of the Third Program Visualization Workshop (PVW 2004), Research Report CS-RR-407*, pages 126–133, Warwick, UK. Department of Computer Science, University of Warwick.

Kölling, M., Quig, B., Patterson, A., and Rosenberg, J. (2003). The BlueJ system and its pedagogy. *Computer Science Education*, 13(4):249–268.

Korhonen, A. (2003). *Visual Algorithm Simulation.* PhD thesis, Helsinki University of Technology. (Tech Rep. No. TKO-A40/03).

Korhonen, A., Laakso, M.-J., and Myller, N. (2009). How does algorithm visualization affect collaboration? Video Analysis of Engagement and Discussions. Accepted to the *5th International Conference on Web Information Systems and Technologies (WEBIST).*

Korhonen, A., Malmi, L., Myllyselkä, P., and Scheinin, P. (2002). Does it make a difference if students exercise on the web or in the classroom? *SIGCSE Bulletin*, 34(3):121–124.

Korhonen, A., Malmi, L., Silvasti, P., Karavirta, V., Lönnberg, J., Nikander, J., Stålnacke, K., and Ihantola, P. (2004). Matrix — a framework for interactive software visualization. Research Report TKO-B 154/04, Laboratory of Information Processing Science, Department of Computer Science and Engineering, Helsinki University of Technology.

Laakso, M.-J., Myller, N., and Korhonen, A. (2009). Comparing learning performance of students using algorithm visualizations collaboratively on different engagement levels. Accepted for publication in *Journal of Educational Technology*

& Society.

Laakso, M.-J., Salakoski, T., Grandell, L., Qiu, X., Korhonen, A., and Malmi, L. (2005). Multi-perspective study of novice learners adopting the visual algorithm simulation exercise system trakla2. *Informatics in Education*, 4(1):49–68.

Lahtinen, S.-P., Sutinen, E., and Tarhio, J. (1998). Automated Animation of Algorithms with Eliot. *Journal of Visual Languages and Computing*, 9(3):337–349.

Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 33:159–174.

Lattu, M., Meisalo, V., and Tarhio, J. (2003). A visualization tool as a demonstration aid. *Computers & Education*, 41(2):133–148.

Lattu, M., Tarhio, J., and 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*, pages 19–32, Corenza, Italy. `http://www.ppig.org/papers/12th-lattu.pdf` (Accessed 2009-04-01).

Lauer, T. (2006). Learner interaction with algorithm visualizations: viewing vs. changing vs. constructing. In *ITICSE '06: Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education*, pages 202–206, New York, NY, USA. ACM.

Lauer, T. (2008a). Reevaluating and refining the engagement taxonomy. In *ITiCSE '08: Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pages 355–355, New York, NY, USA. ACM.

Lauer, T. (2008b). When does algorithm visualization improve algorithm learning? — reviewing and refining an evaluation framework. In Cortesi, A. and Luccio, F., editors, *Proceedings of ACM-IFIP Informatics Education Europe III*, pages 198–208. `http://www.dsi.unive.it/IEEIII/atti/PROCEEDINGS_IEEIII08.pdf` (Accessed 2009-04-01).

Lauer, T. and Myller, N. (2008). Personal communication. discussions and emails.

Lavonen, J., Meisalo, V., and Lattu, M. (2002). Collaborative problem solving in a control technology learning environment, a pilot study. *International Journal of Technology and Design Education*, 12(2):139–160.

Lavonen, J., Meisalo, V., Lattu, M., and Sutinen, E. (2003). Concretising the programming task: a case study in a secondary school. *Computers and Education*, 40(2):115–135.

Lehtinen, E., Hakkarainen, K., Lipponen, L., Rahikainen, M., and Muukkonen, H. (1999). Computer supported collaborative learning: A review. The J.H.G.I. Giesbers Reports on Education 10, Department of Educational Sciences, University on Nijmegen. `http://www.kas.utu.fi/papers/clnet/clnetreport.html` (Accessed 2009-04-01).

Lister, R. (2005). Mixed methods: positivists are from mars, constructivists are from venus. *SIGCSE Bulletin*, 37(4):18–19.

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., Mc-

Cartney, R., Moström, J. E., Sanders, K., Seppälä, O., Simon, B., and Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. *SIGCSE Bulletin*, 36(4):119–150.

Lukka, K. (2003). The constructive research approach. In Ojala, L. and Hilmola, O.-P., editors, *Case study research in logistics. Publications of the Turku School of Economics and Business Administration, Series B 1*, pages 83–101.

Malmi, L., Karavirta, V., Korhonen, A., Nikander, J., Seppälä, O., and Silvasti, P. (2004). Visual algorithm simulation exercise system with automatic assessment: Trakla2. *Informatics in Education*, 3(2):267–288.

Markkanen, J., Saariluoma, P., Sutinen, E., and Tarhio, J. (1998). Visualization and imagery in teaching programming. In Domingue, J. and Mulholland, P., editors, *10th Annual Meeting of the Psychology of Programming Interest Group*, pages 70–73, Knowledge Media Institute, Open University, Milton Keynes, UK.

Mattessich, P. W., Murray-Close, M., and Monsey, B. R. (2004). *Collaboration: What Makes It Work.* Fieldstone Alliance, 2nd edition.

Mayer, R. E. (2001). *Multimedia Learning.* Cambridge University Press, Cambridge, UK.

Mayer, R. E. and Chandler, P. (2001). When Learning is Just a Click Away: Does Simple User Interaction Foster Deeper Understanding of Multimedia Messages? *Journal of Educational Psychology*, 93(2):390–397.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Ben-David Kolikant, Y., Laxer, C., Thomas, L., Utting, I., and Wilusz, T. (2001). A multinational, multi-institutional study of assessment of programming skills of first-year cs students. *SIGCSE Bulletin*, 33(4):125–180.

McDowell, C., Werner, L., Bullock, H. E., and Fernald, J. (2003). The impact of pair programming on student performance, perception and persistence. In *Proceedings of the 25th International Conference on Software Engineering*, pages 602–607. IEEE Computer Society.

McGettrick, A., Boyle, R., Ibbett, R., Lloyd, J., Lovegrove, G., and Mander, K. (2005). Grand challenges in computing: Education–a summary. *The Computer Journal*, 48(1):42–48.

McGuigan, F. J. (1996). *Experimental Psychology Methods of Research.* Prentice Hall, 7th edition.

Mead, G. (1977). *On social psychology. Selected papers.* The University of Chicago Press, Chicago. A. Strauss (Ed.).

Meier, A., Spada, H., and Rummel, N. (2007). A rating scheme for assessing the quality of computer-supported collaboration processes. *International Journal of Computer-Supported Collaborative Learning*, 2(1):63–86.

Meisalo, V., Sutinen, E., and Torvinen, S. (2003). Choosing Appropriate Methods for Evaluating and Improving the Learning Process in Distance Programming Courses. In *Proceedings of the 33rd ASEE/IEEE Frontiers in Education Confer-*

ence (FIE2003), pages T2B–11–16, Boulder, CO, USA.

Mendes, A. J., Gomes, A., Esteves, M., Marcelino, M. J., Bravo, C., and Redondo, M. A. (2005). Using simulation and collaboration in cs1 and cs2. *SIGCSE Bulletin*, 37(3):193–197.

Milne, I. and Rowe, G. (2002). Difficulties in learning and teaching programming–views of students and tutors. *Education and Information Technologies*, 7(1):55–66.

Moreno, A. (2005). The Design and Implementation of Intermediate Codes for Software Visualization. Master's thesis, Department of Computer Science, University of Joensuu, Joensuu, Finland.

Moreno, A., Bednarik, R., and Yudelson, M. (2007a). How to Adapt the Visualization of Programs? In Brusilovsky, P., Grigoriadou, M., and Papanikolaou, K., editors, *Proceedings of Workshop on Personalisation in E-Learning Environments at Individual and Group Level, 11th International Conference on User Modeling*, pages 65–70.

Moreno, A. and Joy, M. S. (2007). Jeliot 3 in a demanding educational setting. *Electronic Notes in Theoretical Computer Science*, 178:51–59.

Moreno, A., Myller, N., Ben-Ari, M., and Sutinen, E. (2004b). Program Animation in Jeliot 3. In *Proceedings of the 9th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2004)*, page 265, Leeds, UK. ACM SIGCSE.

Moreno, A., Myller, N., Sutinen, E., and Ben-Ari, M. (2004a). Visualizing Programs with Jeliot 3. In *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '04)*, pages 373–376, Gallipoli (Lecce), Italy.

Moreno, A., Sutinen, E., Bednarik, R., and Myller, N. (2007b). Conflictive animations as engaging learning tools. In Lister, R. and Simon, editors, *Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007)*, pages 203–206.

Moreno, R. and Mayer, R. E. (2000). Engaging students in active learning: The case for personalized multimedia messages. *Journal of Educational Psychology*, 92(4):724–733.

Myller, N. (2004). Fundamental Design Issues of Jeliot 3. Master's thesis, Department of Computer Science, University of Joensuu, Joensuu, Finland.

Myller, N. (2007). Automatic generation of prediction questions during program visualization. *Electronic Notes in Theoretical Computer Science*, 178:43–49. (Proceedings of the Fourth Program Visualization Workshop).

Myller, N., Bednarik, R., Ben-Ari, M., and Sutinen, E. (2009). Extending the engagement taxonomy: Software visualization and collaborative learning. Accepted for publication in *ACM Transactions on Computing Education* (formerly The ACM Journal on Educational Resources in Computing).

Myller, N., Bednarik, R., and Moreno, A. (2007a). Integrating Dynamic Program Visualization into BlueJ: the Jeliot 3 Extension. In *Proceedings of the 7th IEEE*

International Conference on Advanced Learning Technologies (ICALT'07), pages 505–506.

Myller, N., Laakso, M., and Korhonen, A. (2007b). Analyzing engagement taxonomy in collaborative algorithm visualization. In Hughes, J., Peiris, D. R., and Tymann, P. T., editors, *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE '07)*, pages 251–255, New York, NY, USA. ACM Press.

Myller, N. and Nuutinen, J. (2006). JeCo: Combining program visualization and story weaving. *Informatics in Education*, 5(2):195–206.

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., and Balik, S. (2003). Improving the CS 1 experience with pair programming. In *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, pages 359–362. ACM Press.

Naps, T. L. (2005). JHAVÉ – Addressing the need to support algorithm visualization with tools for active engagement. *IEEE Computer Graphics and Applications*, 25(5):49–55.

Naps, T. L., Eagan, J. R., and Norton, L. L. (2000). JHAVÉ – An Environment to Actively Engage Students in Web-based Algorithm Visualizations. In *Proceedings of the thirty-first SIGCSE technical symposium on Computer Science Education*, pages 109–113, New York, NY, USA. ACM Press.

Naps, T. L. and Grissom, S. (2002). The effective use of quicksort visualizations in the classroom. *Journal of Computing Sciences in Colleges*, 18(1):88–96.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., and Velázquez-Iturbide, J. Á. (2002). Exploring the Role of Visualization and Engagement in Computer Science Education. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, pages 131–152, New York, NY, USA. ACM Press.

Nosek, J. T. (1998). The case for collaborative programming. *Communication of the ACM*, 41(3):105–108.

Paterson, J. H. and Haddow, J. (2007). From classes to code: supporting the transition from design to implementation. In *ITiCSE '07: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, pages 362–362, New York, NY, USA. ACM.

Paterson, J. H., Haddow, J., and Nairn, M. (2006). A design patterns extension for the bluej ide. In *ITICSE '06: Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education*, pages 280–284, New York, NY, USA. ACM.

Patterson, A., Kölling, M., and Rosenberg, J. (2003). Introducing unit testing with bluej. In *ITiCSE '03: Proceedings of the 8th annual conference on Innovation and technology in computer science education*, pages 11–15, New York, NY, USA.

ACM.

Petre, M. (1995). Why Looking Isn't Always Seeing: Readership Skills and Graphical Programming. *Communication of the ACM*, 38(6):55–70.

Petre, M. and Green, T. R. G. (1993). Learning to Read Graphics: Some Evidence that 'Seeing' an Information Display Is an Acquired Skill. *Journal of Visual Languages and Computing*, 4(1):55–70.

Phillip, R., Roy, R., and Sharon, A. (1995). Collaborative learning for computer science students. *Journal of Computers in Mathematics and Science Teaching*, 14(3):377–389.

Piaget, J. (1980). *The Constructivist approach.* Foundation Archives Jean Piaget, Geneva.

Pierson, W. C. and Rodger, S. H. (1998). Web-based animation of data structures using JAWAA. *SIGCSE Bulletin*, 30(1):267–271.

Price, B. A., Baecker, R. M., and Small, I. S. (1993). A Principled Taxonomy of Software Visualization. *Journal of Visual Languages & Computing*, 4(3):211–266.

Ragonis, N. and Ben-Ari, M. (2005a). A long-term investigation of the comprehension of oop concepts by novices. *Computer Science Education*, 15(3):203–221.

Ragonis, N. and Ben-Ari, M. (2005b). On understanding the statics and dynamics of object-oriented programs. *SIGCSE Bulletin*, 37(1):226–230.

Ratcliffe, M. B. and Thomas, L. A. (2004). Understanding our students: Incorporating the results of several experiments into a student learning environment. In *16th Workshop of the Psychology of Programming Interest Group*, pages 10–20.

Rhodes, P., Kraemer, E., and Reed, B. (2006). The importance of interactive questioning techniques in the comprehension of algorithm animations. In *Proceedings of the ACM Symposium on Software Visualization (SOFTVIS 2006)*, pages 183–184, Brighton, UK.

Ridgway, M., Ratcliffe, M., and Ellis, W. (2003). Vortex — enhancing the pedagogy in software development education. *Proceedings of the American Society for Information Science and Technology*, 40(1):542.

Robins, A., Rountree, J., and Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2):137–172.

Roman, G.-C., Cox, K. C., Wilcox, D., and Plun, J. Y. (1992). Pavane: a System for Declarative Visualization of Concurrent Computations. *Journal of Visual Languages and Computing*, 3(2):161–193.

Roschelle, J. (1992). Learning by collaborating: Convergent conceptual change. *The Journal of the Learning Sciences*, 2(3):235–276.

Roschelle, J. (1996). Designing for cognitive communication: Epistemic fidelity or mediating collaborative inquiry. In Day, D. L. and Kovacs, D. K., editors, *Computers, Communication & Mental Models*, pages 13–25. Taylor & Francis, London.

Rößling, G. and Freisleben, B. (2002). Animal: A system for supporting multiple roles in algorithm animation. *Journal of Visual Languages and Computing*, 13(3):341–354.

Rößling, G. and Häussge, G. (2004). Towards tool-independent interaction support. In Korhonen, A., editor, *Proceedings of the Third International Program Visualization Workshop*, pages 110–117, Warwick, England.

Rößling, G. and Naps, T. L. (2002). A testbed for pedagogical requirements in algorithm visualizations. In *Proceedings of the Innovation and Technology in Computer Science Education (ITiCSE'02)*, pages 96–100. ACM Press.

Sajaniemi, J. and Kuittinen, M. (2005). An experiment on using roles of variables in teaching introductory programming. *Computer Science Education*, 15(1):59–82.

Scaife, M. and Rogers, Y. (1996). External cognition: how do graphical representations work? *International Journal of Human-Computer Studies*, 45(2):185–213.

Scardamalia, M., Bereiter, C., McLean, R., Swallow, J., and Woodruff, M. (1989). Computer supported intentional learning enviroments. *Journal of Educational Computing research*, 5(1):51–68.

Seppälä, O., Malmi, L., and Korhonen, A. (2006). Observations on student misconceptions — a case study of the build-heap algorithm. *Computer Science Education*, 16(3):241–255.

Slavin, R. (1995). *Cooperative learning: Theory research and practice.* Ally & Bacon, Boston, MA.

Spada, H., Meier, A., Rummel, N., and Hauser, S. (2005). A new method to assess the quality of collaborative process in cscl. In Koschmann, T., Suthers, D., and Chan, T. W., editors, *Computer Supported Collaborative Learning 2005: The Next 10 Years!*, pages 622–631, Mahwah, NJ. Lawrence Erlbaum.

Suthers, D., Dwyer, N., Medina, R., and Vatrapu, R. (2007a). A framework for eclectic analysis of collaborative interaction. In Chinn, C., Erkens, G., and Puntambekar, S., editors, *The Computer Supported Collaborative Learning (CSCL) Conference 2007*, pages 694–703, New Brunswick. International Society of the Learning Sciences.

Suthers, D., Toth, E., and Weiner, A. (1997). An integrated approach to implementing collaborative inquiry in the classroom. In *Proceedings of the 2nd International Conference on Computer Supported Collaborative Learning (CSCL'97)*, pages 272–279, Toronto.

Suthers, D., Vatrapu, R., Medina, R., Joseph, S., and Dwyer, N. (2007b). Conceptual representations enhance knowledge construction in asynchronous collaboration. In Chinn, C., Erkens, G., and Puntambekar, S., editors, *The Computer Supported Collaborative Learning (CSCL) Conference 2007*, pages 704–713, New Brunswick. International Society of the Learning Sciences.

Suthers, D., Vatrapu, R., Medina, R., Joseph, S., and Dwyer, N. (2008). Beyond threaded discussion: Representational guidance in asynchronous collaborative learning environments. *Computers & Education*, 50(4):1103–1127.

Suthers, D. D. (1999). The effects of representational bias on collaborative inquiry. In *Proceedings of the 8th International Conference on Human-Computer Interaction: Ergonomics and User Interfaces - Volume I*, pages 362–366, Hillsdale, NJ, USA. Lawrence Erlbaum Associates Inc.

Suthers, D. D. and Hundhausen, C. D. (2003a). An experimental study of the effects of representational guidance on collaborative learning processes. *Journal of the Learning Sciences*, 12(2):183–219.

Suthers, D. D., Hundhausen, C. D., and Girardeau, L. E. (2003b). Comparing the roles of representations in face-to-face and online computer supported collaborative learning. *Computers & education*, 41(4):335–351.

Sutinen, E., Tarhio, J., Lahtinen, S.-P., Tuovinen, A.-P., Rautama, E., and Meisalo, V. (1997). Eliot – an Algorithm Animation Environment. Report A-1997-4, Department of Computer Science, University of Helsinki, Helsinki, Finland. `http://www.cs.helsinki.fi/TR/A-1997/4/A-1997-4.ps.gz` (Accessed 2009-04-01).

Sutinen, E., Tarhio, J., and Teräsvirta, T. (2003). Easy Algorithm Animation on the Web. *Multimedia Tools and Applications*, 19(2):179–184.

Teague, D. and Roe, P. (2008). Collaborative learning: towards a solution for novice programmers. In *ACE '08: Proceedings of the tenth conference on Australasian computing education*, pages 147–153, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

Teasley, S. (1997). Talking about reasoning: How important is the peer in peer collaboration. In Resnick, L., Säljö, R., Pontecorvo, C., and Burge, B., editors, *Discourse, Tools and Reasoning: Essays on Situated Cognition*, pages 361–384. Springer, New York.

Urquiza-Fuentes, J. and Velázquez-Iturbide, J. Á. (2007). An Evaluation of the Effortless Approach to Build Algorithm Animations with WinHIPE. *Electronic Notes in Theoretical Computer Science*, 178:3–13. (Proceedings of the Fourth Program Visualization Workshop).

Valdivia, R. and Nussbaum, M. (2007). Face-to-face collaborative learning in computer science classes. *International Journal of Engineering Education*, 23:434–440(7).

Vygotsky, L. (1978). *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press, Cambridge, MA.

Williams, L., Kessler, R. R., Cunningham, W., and Jeffries, R. (2000). Strengthening the case for pair programming. *IEEE Software*, 17(4):19–25.

Wills, C., Deremer, D., McCauley, R., and Null, L. (1999). Studying the use of peer learning in the introductory computer science curriculum. *Computer Science Education*, 9:71–88.

Xinogalos, S., Satratzemi, M., and Dagdilelis, V. (2007). Teaching java with bluej: a two-year experience. In *ITiCSE '07: Proceedings of the 12th annual SIGCSE* *conference on Innovation and technology in computer science education*, pages 345–345, New York, NY, USA. ACM.

Original Publications

P1.

1

Myller, N., Bednarik, R., Ben-Ari, M., and Sutinen, E. (2009). Extending the engagement taxonomy: software visualization and collaborative learning**.** *ACM Transactions on Computing Education*, 9(1), Article 7.

Extending the Engagement Taxonomy: Software Visualization and Collaborative Learning

NIKO MYLLER, ROMAN BEDNARIK, and ERKKI SUTINEN
University of Joensuu
and
MORDECHAI BEN-ARI
Weizmann Institute of Science

As collaborative learning in general, and pair programming in particular, has become widely adopted in computer science education, so has the use of pedagogical visualization tools for facilitating collaboration. However, there is little theory on collaborative learning with visualization, and few studies on their effect on each other. We build on the concept of the *engagement taxonomy* and extend it to classify finer variations in the engagement that result from the use of a visualization tool. We analyze the applicability of the taxonomy to the description of the differences in the collaboration process when visualization is used. Our hypothesis is that increasing the level of engagement between learners and the visualization tool results in a higher positive impact of the visualization on the collaboration process. This article describes an empirical investigation designed to test the hypothesis. The results provide support for our extended engagement taxonomy and hypothesis by showing that the collaborative activities of the students and the engagement levels are correlated.

Categories and Subject Descriptors: K.3.2 [**Computer Science Education**]: Computer & Information Science Education—*Computer Science Education*

General Terms: Algorithms, Experimentation, Human Factors

Additional Key Words and Phrases: Program visualization, collaborative learning, engagement taxonomy

The research of N. Myller and R. Bednarik has been funded by the East Finland Graduate School in Computer Science and Engineering (ECSE). A part of this paper was written while N. Myller was a visiting lecturer at the Tumaini University, Iringa University College, Iringa, Tanzania under the 5ARTS+ project funded by CIMO.
Authors' addresses: N. Myller, R. Bednarik, and E. Sutinen, Department of Computer Science and Statistics, University of Joensuu, P.O. Box 111, FI-80101 Joensuu, Finland; email: nmyller@cs.joensuu.fi; M. Ben-Ari, Department of Science Teaching, Weizmann Institute of Science, Rehovot 76100, Israel.

ACM Reference Format:
Myller, N., Bednarik, R., Sutinen, E., and Ben-Ari, M. 2009. Extending the engagement taxonomy: Software visualization and collaborative learning. ACM Trans. Comput. Educ. 9, 1, Article 7 (March 2009), 27 pages. DOI = 10.1145/1513593.1513600.
http://doi.acm.org/10.1145/1513593.1513600.

1. INTRODUCTION

When *algorithm visualization (AV)* and *program visualization (PV)*[1] were first introduced more than two decades ago, they seemed to be a silver bullet that could solve difficult problems related to the teaching and learning of programming, data structures and algorithms. However, the mixed results of empirical evaluations have made the benefits of visualization tools as teaching and learning aids questionable [Hundhausen et al. 2002]. Therefore, researchers have begun to seek explanations for the mixed results in order to discover the conditions under which visualization tools can actually achieve improvements in learning.

In a meta-analysis of the research on AV, Hundhausen et al. [2002] concluded that the activities performed by students and their engagement seem to be more important than the subject content or the graphic elements of the visualization. The findings led to the analysis of different engagement levels between the user and the visualization tool, resulting in the *engagement taxonomy (ET)* described by Naps et al. [2002]. The main assumption of the taxonomy is that the level of engagement between the user and the visualization affects the learning of the individual student. Since its introduction, the ET has guided the research and development of SV tools, and several studies have utilized the framework [Grissom et al. 2003; Naps and Grissom 2002].

Collaborative learning and pair programming have become accepted and popular methods in computer science education [Hundhausen and Brown 2008; McDowell et al. 2006; Simon et al. 2004; Nagappan et al. 2003; Hundhausen 2002; Williams et al. 2000]; as the use of visualization tools increases [Naps et al. 2002], they appear more and more often in situations of collaborative learning. This combination introduces new challenges and possibilities that are different from the ones related to individual learning with visualization. On the one hand, successful collaboration requires the communication of knowledge and new ideas between group members, as well as the coordination of joint work, but it is not clear how visualization affects these issues [Suthers and Hundhausen 2003]. On the other hand, visualization tools themselves can create a context for collaboration by providing a shared external representation that can initiate negotiations of meanings; they can also become a reference point for explaining ideas or resolving misunderstandings. The mutual influence of visualization and collaboration on each other is likely to be relevant for their joint analysis through means such as the engagement

[1]We will use *software visualization (SV)* to refer to both of these subfields.

taxonomy; therefore, it is unlikely that the ET for collaborative learning will be same as it is for individual learning.

Visualization tools that are used during collaboration can be divided into two categories: *information visualization* such as concept maps or SV tools that visualize programs or data structures, and *augmenting visualization* such as social and group awareness software. Augmenting visualization tools are known to enhance the process and the outcomes of the collaboration [Janssen et al. 2007]. In this article, we concentrate on information visualization, particularly on SV, because the effects of its content and form on collaboration have not been investigated in depth. There are few theories of collaborative learning that apply to information visualizations and few tools that support the collaborative learning [Suthers and Hundhausen 2003; Suthers et al. 2003]. Thus, users have little guidance from the research literature as to how to adjust the use of a tool meant for an individual to the different needs of collaborative settings [Bryant et al. 2005].

In this paper, we first review literature related to the collaborative use of visualization tools (Section 2). We then extend the engagement taxonomy with 1) new levels to an *extended engagement taxonomy (EET)* that identifies finer levels of distinctions in the engagement in both individual and collaborative learning (Section 3.1) and 2) a hypothesis that takes into account aspects of collaborative learning process (Section 3.2).

Our hypothesis is that the higher the level of engagement between learners and the visualization tool, the higher is the positive impact of the visualization on the collaboration process. To test this hypothesis, we present an empirical study, in which we investigated the activities of groups of students at different engagement levels as supported by two visualization tools (Section 4). In the study, we analyzed the interactions between students, and between the students and a visualization. Section 5 presents the results and provides evidence for our hypothesis. In the final section, we discuss the implications of our findings on the future research and development of collaborative learning with software visualization.

2. RELATED WORK

This section is an integrated survey of the relevant previous work on visualization, collaborative learning and the engagement taxonomy.

2.1 Software Visualization and the Engagement Taxonomy

In an attempt to describe the mixed results of previous research on SV in education, the engagement taxonomy was introduced by Naps et al. [2002]. Its purpose is to describe the different forms of engagement that a visualization tool can promote, and to provide testable hypotheses about the use of visualizations in the teaching and learning of computer science. The central idea of the taxonomy is that higher-level engagement between learner and the visualization results in better learning outcomes. The ET consists six levels of engagement between the user and the visualization (see Table I).

Table I. The Engagement Taxonomy

No viewing	There is no visualization to be viewed.
Viewing	The visualization is only looked at without any other form of engagement.
Responding	Learners are presented with questions related to the visualization.
Changing	Modification of the visualization is allowed, for example, by varying the input data set.
Constructing	Learners are expected to create their own visualization of a program or an algorithm.
Presenting	Learners present visualizations to others for feedback and discussion.

When there is no visualization to look at, the engagement is, of course, at its lowest level. Passive viewing of a visualization seems to improve learning outcomes very little, if at all, even when compared with the no viewing level [Hundhausen et al. 2002; Naps et al. 2002; Naps 2005]. One can conclude that there should be an active component in the learning process in order to enhance learning with visualization. That is, the viewing of a visualization should be combined with activities at the higher levels of engagement: responding, changing, constructing or presenting. To our knowledge, there are few empirical comparisons between these forms of active engagement on learning outcomes [Naps et al. 2002; Naps 2005]. In light of current research, the taxonomy forms a three-level hierarchy: no engagement, passive engagement, and active engagement [Naps 2005].

The ET has been used in the development of AV tools, and studies have validated its applicability [Grissom et al. 2003; Naps and Grissom 2002; Myller et al. 2007b].

Other studies—although not using the ET—have shown that visualizations enhance learning; for example, Ben-Bassat Levy et al. [2003] found that students who actively used the Jeliot program animation system improved their learning results compared with a control group that did not use Jeliot. Also relevant is the research in educational psychology and multimedia learning, which has found a positive effect of the interactivity in multimedia on learning outcomes [Evans and Gibbons 2007]. These studies have concentrated on changes in the learning outcomes when visualization is used. In this article, we extend the scope of such research in order to investigate the impact of the engagement with visualization tools on the learning process.

The effects of visualization on the learning process have been also researched, although not in a collaborative environment. For example, Ebel and Ben-Ari [2006] showed that program visualization increases the attention of students to the material being taught. We believe that this could also hold in a collaborative environment: as students' attention increases, they will be able to concentrate better on the collaborative activities, but it should be tested in the future experiments.

2.2 Use of Visualizations in Collaborative Learning

The work of Roschelle [1996] is considered seminal for the whole field of *computer-supported collaborative learning (CSCL)*. Roschelle developed the *envisioning machine*, a software tool for studying mechanics that students use to manipulate simple diagrams related to velocity and acceleration. He investigated how pairs of students used the tool, and he analyzed the learning outcomes as well as the processes that led to those outcomes. He recognized that learning tools for collaboration should be designed to support communication, rather than merely to present the underlying model as accurately as possible. Roschelle [1996] gives a number of guidelines in order to achieve this goal; the final one is "one should design activities which actively engage students in doing and encounter [sic] meaningful experiential feedback as a consequence of their actions" (p. 14). The analysis of the interaction between the external presentation and users was also identified as a key research area by Scaife and Rogers [1996]. The work of Roschelle [1996] and Scaife and Rogers [1996] reflects the idea that the engagement with visualizations affects collaborative learning, and we build on this in the article.

Suthers and Hundhausen [2003] compared the effects of different representations (matrices, graphs, text) when students collect and analyze data, form hypotheses and investigate their evidential relations, both in a face-to-face and in a distance context [Suthers et al. 2003]. They found that there were differences in the guidance that different representations give to the collaboration, especially to discussions, and that the different learning situations (face-to-face or distance) affect the usage of the representations [Suthers et al. 2003]. However, they did not find differences in the performance of the students, but only in the way the students used and discussed the representations. It could be argued that the differences in the study process should have an effect on students' learning only in a long run, and therefore were not detectable in the laboratory setting.

The research described in this section shows that visualization and the kinds of interactions it drives have an effect on the collaboration process and could affect the collaboration outcomes.

2.3 Research on Software Visualization in Collaborative Learning

Although a plethora of software visualization tools have been developed and empirical studies carried out, there have been only a few tools and studies relating to the collaborative use of SV.

Myller et al. [2007b] studied learning outcomes after students had collaboratively learned about the concept of binary heap with the help of either animation (ET level: viewing) or algorithm simulation (ET level: changing). Although they did not find statistically significant differences in the performance between the groups, the groups that used algorithm simulations performed consistently better in a post-test, compared with the groups that viewed just the animations. In a replication of the study, a statistically significant difference was found in favor of the algorithm simulation group, a finding that supports the applicability of ET in the context of collaborative learning with visualization [Laakso et al. 2008].

Hundhausen [2002] studied the collaborative aspects of AV construction and presentation, and concluded—as did Roschelle [1996]—that the fidelity of the visualization can be compromised in favor of meaningful interactions between students. This led into the development of ALVIS, a visualization tool that supports construction and presentation of AVs [Hundhausen and Brown 2007]. In an experiment, they compared ALVIS as a tool to writing programs for algorithms to a text editor, and then the use of ALVIS for visualization construction and presentation to simple art supplies [Hundhausen and Brown 2008, 2005]. Students worked in pairs and were asked to write an algorithm in the SALSA language supported by ALVIS, construct a visualization of that algorithm, and present it to the instructor and the other students. It was found that pairs of students who used ALVIS (EET level: viewing, constructing) concentrated more on the solution, spent less time unproductively, and needed less help from the teaching assistant; in addition, they developed better code than pairs of students who used a text editor (EET level: no viewing).

Hübscher-Younger and Narayanan [2003] developed a Web-based system that allowed students to publish their own algorithm representations (text, pictures, animations, multimedia) and discuss them on the Web. They concluded that students who actively participated in this activity achieved higher grades than the passive students who might have only viewed and commented the other students' presentations.

Jehng and Chan [1998] designed and evaluated a distributed visual learning environment for LISP-LOGO that supported collaborative learning. The results showed that students who learned collaboratively, either face-to-face or at a distance, outperformed individual learners in program generation tasks, but that all groups performed equally well in program evaluation and completion tasks. This shows that while collaborative visual learning can be more beneficial compared to individual learning, the improvement can depend on the specific learning task.

Hundhausen [2005] proposed the *communicative dimensions (CD)* framework as a theory on the use of visualizations as communication tools.[2] CD describes the aspects of visualization environments that have an effect on communication between its users in six dimensions: programming salience, provisionality, story content, modifiability, controllability, referencability. While the CD framework is concerned solely with the properties of a visualization tool, the ET framework concerns itself with the interaction between users and visualizations. We think it is very important to understand this interaction and its different levels in order to make tools that support successful collaboration.

2.4 Successful Collaboration Processes

The notion of a successful collaboration process is controversial and not easy to define.

Meier et al. [2007] used a combination of top-down and bottom-up approaches in order to form a description of a successful collaboration. They carried out a comprehensive review of literature, focusing on the aspects of

[2]CD was inspired by the Cognitive Dimensions framework for analyzing individual user interaction with software tools [Green and Petre 1996].

Table II. Aspects and Dimensions of a Successful Collaboration Process [Meier et al. 2007]

Aspect	Dimension
Communication	1) Sustaining mutual understanding
Communication	2) Dialogue management
Joint information processing	3) Information pooling
Joint information processing	4) Reaching consensus
Coordination	5) Task division
Coordination	6) Time management
Coordination	7) Technical coordination
Interpersonal relationship	8) Reciprocal interaction
Motivation	9) Individual task orientation

a successful collaboration (top-down). In addition, they used a data-driven approach to create dimensions from empirical data under each aspect (bottom-up) [Spada et al. 2005]. The study identified five aspects and under them nine dimensions that describe various perspectives of a successful collaboration (see Table II). Furthermore, Meier et al. [2007] used these dimensions in a rating scheme, which they validated with empirical data. The results showed that the high scores on the dimensions of the collaboration process correlate strongly with good results of the collaboration.

The problem with their rating scheme is the difficulty of achieving high inter-rater reliabilities in the ratings. However, this does not mean that the dimensions and aspects of the successful collaboration are not reasonable, and do not reflect the qualities of a successful collaboration. It just means that it is difficult to judge how they appear in the collaboration process.

Teasley [1997] investigated the importance of discussions during collaboration and showed that the amount of discussion is an important part of successful collaboration. However, the amount of transactive reasoning in discussions seems to be an even stronger factor for successful collaboration. Transactive reasoning means talking about one's own thinking process (i.e., reasoning) or one's understanding of the partners' thinking processes [Berkowitz and Gibbs 1983]. In our context, this would mean, for example, that a student talks about what different components of the visualizations mean to him/her or what will happen next in the visualization. Teasley [1997] also showed that a partner is not necessary for transactive reasoning to happen, although the likelihood for it to happen increases when a partner with similar knowledge level is available.

We can summarize these results as showing that successful collaboration requires interaction (dimensions: 1, 2, 4, 8 and Teasley's research), coordination (dimensions: 3, 5, 6, 7) and motivation (dimension: 9).

As discussed in the previous section, Ebel and Ben-Ari [2006] have shown that program visualizations have positive affective effects (i.e., lengthened attention) and anecdotal evidence exists that animation increases students' motivation [Naps et al. 2002]. In addition, Janssen et al. [2007] have showed that augmenting visualizations support the coordination of the collaboration. In this article, we will study how visualization tools affect the interactions in collaboration.

Table III. The Extended Engagement Taxonomy

No viewing (*)	There is no visualization to be viewed but only material in textual format. For example, the students are reviewing the source code without modifying it or they are looking at the learning materials.
Viewing (*)	The visualization is viewed with no interaction. For example, the students are looking at the visualization or the program output.
Controlled viewing	The visualization is viewed and the students control the visualization, for example by selecting objects to inspect or by changing the speed of the animation. This has been deemed important, for instance by Rößling and Naps [2002].
Entering input	The student enters input to a program or parameters to a method before or during their execution.
Responding (*)	The visualization is accompanied by questions which are related to its content.
Changing (*)	Changing of the visualization is allowed during the visualization, for instance, by direct manipulation.
Modifying	Modification of the visualization is carried out before it is viewed, for example, by changing source code or an input set.
Constructing (*)	The visualization is created interactively by the student by construction from components such as text and geometric shapes.
Presenting (*)	Visualizations are presented and explained to others for feedback and discussion.
Reviewing	Visualizations are viewed for the purpose of providing comments, suggestions and feedback on the visualization itself or on the program or algorithm.

3. EXTENDING THE ENGAGEMENT TAXONOMY

3.1 Engagement Levels

The categories of the ET are primarily based on work in AV research, and thus reflect the types of engagement support that are found in AV tools. However, other types of engagement are supported in *PV* tools, and we find it necessary to extend the ET framework in order to capture these differences. Whereas in AV the interaction of the student with the software is more or less restricted to modifications of the visualization itself to the extent allowed by the tool, in PV the opportunities for engagement include both interactive input and, more importantly, the ability to modify the source code that is the basis for the visualization. Consider the software tools we used (see below): In Jeliot, dynamic animations are generated automatically whenever the source code is changed, and BlueJ is based upon interactive calls of methods of a Java class that are regenerated immediately upon modification of a program.

These considerations guided the development of our *extended engagement taxonomy (EET)* shown in Table III. The levels marked with (*) belong to the original ET, although some definitions were slightly modified. Note, in particular, that *changing* in the ET has been divided into two categories, *changing* and *modifying*. We have added new categories: *controlled viewing*, *entering input*, and *reviewing*. Reviewing is different from presenting in that there is not a specific presenter of the visualization; this category (based on a proposal of Oechsle and Morth [2007]) was added for completeness, although it did not occur in our experimental data.

3.2 Linking Engagement to Collaboration Process

Currently, the ET and EET can be used to generate testable hypotheses only about learning outcomes in individual learning. Based on the evidence from Hundhausen and Brown [2008], Myller et al. [2007b] and Laakso et al. [2008], the learning outcome predictions hold also in collaborative learning with visualization. Although the learning results are important, the process leading to them needs to be studied as well, especially in collaborative learning, because in that context, visualization can affect both inter- and intrapersonal learning.

Our goal is to describe how the engagement level affects the (collaborative) learning process. We propose a hypothesis that extends the applicability of EET to collaborative learning with visualizations, with a special emphasis on the collaboration process: the higher the level of engagement between the collaborators and the visualization, the higher the increase in communication and collaboration during the collaborative learning process. This hypothesis builds on the work of Roschelle [1996], Naps et al. [2002], Suthers and Hundhausen [2003], and Hundhausen [2005] as discussed in Section 2, by explicating the connection between engagement and collaboration.

Although there is no previous research that investigated the same hypothesis, there is indirect supporting evidence that was obtained in another study. In the experiment described in Hundhausen and Brown [2008] (see Section 2.3), the collaborative use of visualization at the higher engagement level enhanced the learning process, because pairs of students working with ALVIS held more discussions with each other and less with the teaching assistant (i.e., they needed less help from outside the group), they worked more on the solution, and they had fewer unproductive periods. This provides initial support for our hypothesis by showing in what ways the higher engagement level might enhance collaboration. In order to further evaluate the validity and applicability of the hypothesis, we carried out an empirical study that tests it explicitly.

4. RESEARCH METHODOLOGY

4.1 The Research Setting

In order to verify the hypothesis, we carried out a causal-comparative study in order to understand how the use of visualization tools at different levels of engagement and the collaboration process are correlated. The *causal-comparative method* [Gall et al. 2006] was selected because in the study we are observing both the dependent and independent variables, and could not control the independent variable, because we wanted to maintain the high ecological validity. This method is used when the independent variable cannot be controlled (e.g., independent variable is gender or in our case the observed engagement level of the visualization). A causal-comparative study cannot prove causality, but it can show that correlation between the dependent and independent variables exists. Because our hypothesis is about finding a positive correlation between the engagement levels and students collaborative activities, this is a reasonable methodology to be used in the study.

The study was carried out in an introductory programming course at the University of Joensuu during the autumn of 2005. The course contained 40 hours of lectures (two-hour lectures twice a week for ten weeks), and two-hour recitation sessions every week, where students presented their solutions to the assignments. Every week students took part in a compulsory two-hour computer laboratory session, where they solved exercises with the help of an instructor. Three of the sessions each week were taught by one instructor (*I1*) and two sessions by another (*I2*). One of the instructors was the course lecturer. Neither the lecturer nor the other instructor knew the purpose of the study and were not associated with it in any way.

We investigated the use of the *BlueJ* [Kölling et al. 2003], an educational development environment, and *Jeliot 3* [Moreno et al. 2004], a program animation tool, on different levels of engagement during those laboratory sessions. We were not trying to compare the tools, but rather to analyze how the differing levels of engagement promoted by the tools affected the collaboration.

Initially, we planned to use a between-subject design, and, therefore, one of the sessions of each instructor was randomly selected to belong to a control condition using only BlueJ, while the other three sessions formed the treatment groups using both Jeliot 3 and BlueJ. We needed to abandon this design because a large number of students dropped the course a few weeks before the final exam; as a result, the groups became biased and we could not make a proper comparison of the learning processes and learning outcomes. Thus, we decided to use the the level of engagement as the independent variable—regardless of which tool was being used—and the original division into treatment and control groups became superfluous. We used data only from the treatment groups in order to get data when both Jeliot 3 and BlueJ were used by the same groups.

In order to control for the differences between the two instructors, we analyzed only those groups that were taught by a single instructor (i.e., *I1*). To control for a learning effect (where the behavior of the students would change as they became more familiar with the tools), we analyzed data from a single week near the beginning of the course.

4.2 Participants

There were a total of five sessions of 20 students each participating in the compulsory laboratory sessions weekly. The total number of the students who gave their consent to participate in the research was 89. Those who did not agree to participate in the study worked in small groups where no data collection was carried out.

The programming course was primarily taught to first year computer science majors. However, a significant proportion (about 60%) of the students taking the course were students majoring in other subjects who studied computer science as a minor. Additionally, there were major students from previous years who had not yet passed the course. Since the current study is not a controlled experiment, randomization is not necessary. We excluded two students who had very extensive programming experience. Otherwise, a post-hoc analysis showed that the remaining students had similar background knowledge and relatively little previous experience in programming.

Because we only analyzed sessions taught by instructor (*I1*) and excluded the original control session, we, altogether, analyzed data from two sessions, otherwise, ten groups of 3–4 students each, a total of 39 students.

4.3 Materials

During the laboratory sessions students were presented with exercises related to the topics of the course. There were no mini-lectures in the beginning of the session, but the exercises were related to the lectures that were given during the same or previous week to the whole course. The teacher handed out the exercises at the beginning of the session and circled around to help out the students. Only if the instructor spotted that several groups had the same misconception or were stuck on the same issue, the teacher went to the front of the class and announced the issue to all the groups and briefly explained it. When the students thought that they were ready with an exercise, they summoned the teacher to check it.

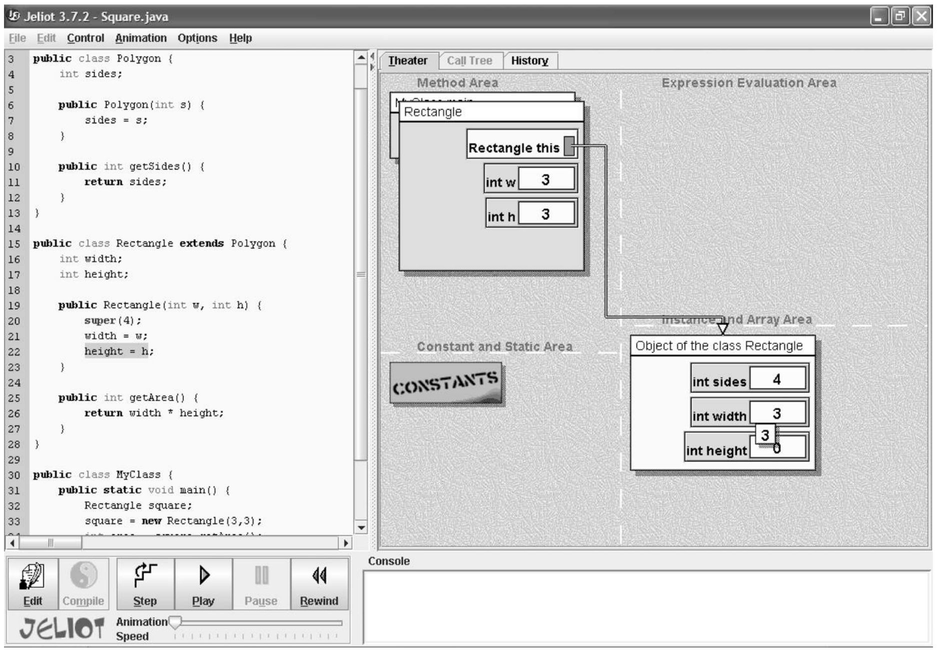
The exercises varied from program construction and modification to debugging. For example, students were given a program code and told what it was supposed to do and they needed to check if the program did what was expected and if it did not, they needed to correct the program. In another exercise, the students were given a skeleton of a program and asked to fill in the missing parts or to create an accompanying class that enabled the program to work as expected. The exercises that were solved in groups were purposefully more difficult than the ones solved individually, because pilot studies indicated that if one of the students could solve the exercise independently, there was neither collaboration nor communication between the students.

4.4 Visualization Tools

Students used both BlueJ and Jeliot during the laboratory sessions. Both tools have proved to be effective in improving the learning of elementary computer science and programming [Ben-Bassat Levy et al. 2003; Haaster and Hagan 2004; Ragonis and Ben-Ari 2005].

The user interface of Jeliot 3 is illustrated in Figure 1. The source code editor is in the left-hand pane, while the right-hand pane is used to display the visualization. VCR-like buttons to control the visualization are located in the lower left corner. Fully dynamic animation of the data and control flow of the program is displayed, including method calls, object construction, and expression evaluation. The animation is created automatically from the source code, so that the student needs only to learn to use the control buttons for the visualization.

Figure 2 shows the user interface of BlueJ. The class diagram is shown in the middle of the window. The student can interactively instantiate an object of a class by right-clicking on the class and selecting the constructor from a popup


```
public class Polygon {
    int sides;

    public Polygon(int s) {
        sides = s;
    }

    public int getSides() {
        return sides;
    }
}

public class Rectangle extends Polygon {
    int width;
    int height;

    public Rectangle(int w, int h) {
        super(4);
        width = w;
        height = h;
    }

    public int getArea() {
        return width * height;
    }
}

public class MyClass {
    public static void main() {
        Rectangle square;
        square = new Rectangle(3,3);
```

Fig. 1. User interface of Jeliot 3.

menu. The objects are then shown at the bottom of the screen as red icons, and the methods of an object can be interactively invoked by right-clicking and selecting the method from a pop-up menu. Parameters of constructors and method calls are also entered interactively.

In the experiment, Jeliot was used as a BlueJ plugin [Myller et al. 2007a] that performs code synchronization between the tools. This allowed students to freely switch between the two tools.

BlueJ and Jeliot were both introduced in the laboratory session of the first week. Students used them individually during the first session so that they were familiar with the tools before the tools were used collaboratively.

4.5 Procedure

There were a total of ten laboratory sessions for each class; during four of them—the 2nd, 4th, 6th, and 8th—students worked in small groups. The students were randomly assigned (by a computer program) to small groups of three or four students, and the membership in the groups was unchanged throughout the course. Due to the high drop-out rate in the course, the number of students in some groups became too low and the groups needed to be merged in the second half of the course. Thus, the materials from the week eight are not comparable to the materials of the other weeks because the groups are not the same.

The learning process in the groups was filmed with video cameras, one for each small group. The filming was done by the first author who was not associi-

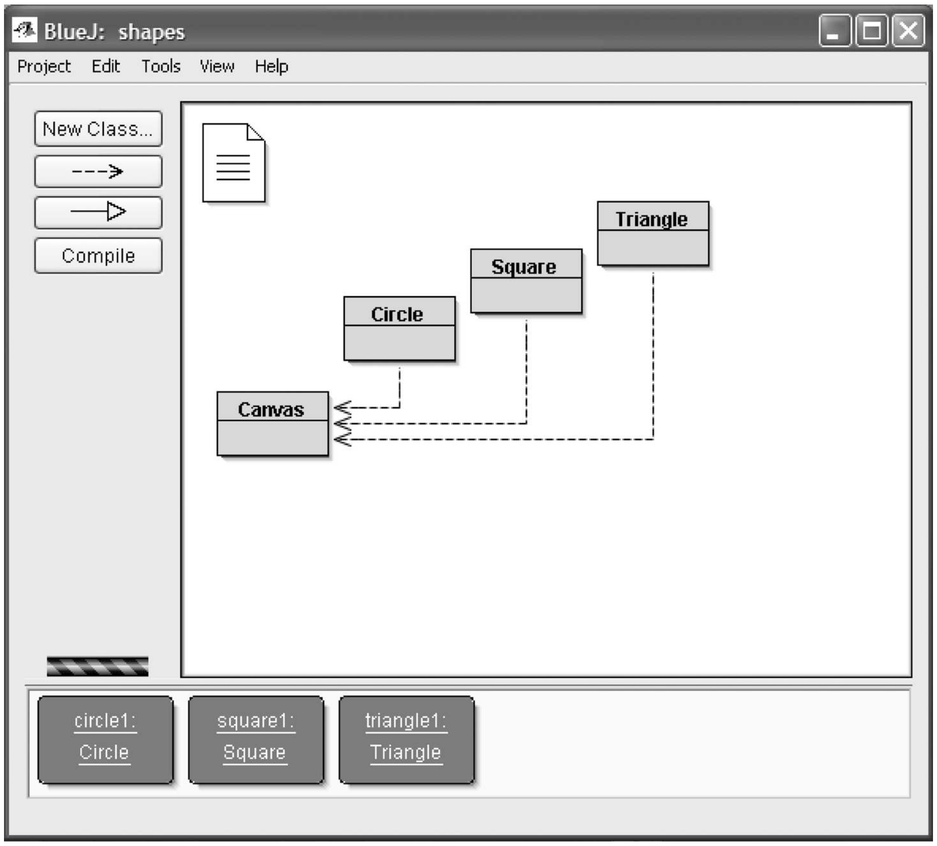


Fig. 2. User interface of BlueJ.

ated with the course. The camera was positioned so that students' movements, facial expressions, and the screen could be recorded. Although minimally invasive, this setting posed some problems in the recordings, because students moved during the lesson and the video cameras could not be always adjusted so that all students' faces were recorded.

Because BlueJ was the primary tool of the course both during the lectures and during the laboratory sessions, the instructor was advised to encourage the students to use Jeliot so that both tools would be used evenly. Since the instructor used both tools as necessary in different exercises, the students were exposed to both tools and could autonomously make decisions when to use each of the tools.

4.6 Data Analysis

To analyze how visualization affected the students' learning processes, we repeatedly viewed those parts of the video materials that contained episodes relating to the execution of a program using one of the tools. These episodes were characterized by extensive use of the visualization capabilities of the tools. Because of the large amount of video-taped material (1.5 hours per small group per week), we randomly sampled 30 two-minute-long episodes. A similar number of episodes were sampled from each group, as were a similar number of episodes using each of the tools. As mentioned above, the analysis was restricted to the sessions of the second week.

In the analysis of the data, we classified each five-second segment of the video according to several classification schemes in order to analyze the level of engagement, and the behavior and the discussions of the students. We used the extended engagement taxonomy (EET) as the independent variable to capture the changes in the level of engagement during an episode in which collaborative learning took place. The EET levels were presented in Table III. Each segment was classified as belonging to a single EET level; in the case where several engagement levels were applicable, the segment was classified at the level that lasted the longest time, or if the times were the same, at the level that was higher in the EET.

The dependent variables were aspects of the students' behavior and discussions; we classified the activities (Section 4.6.1), the discussion contents (Section 4.6.2) and transactive reasoning (Section 4.6.3). The differences found in these classifications were analyzed using the statistical tests Cohen's κ and χ^2 test of independence. To adhere to the limitation of χ^2 test the categories of data with counts less than ten were excluded from the analysis.

4.6.1 *Activities.* We adopted and modified the activity categories of Hundhausen and Brown [2008] to classify the five-second clips of the episodes (see Table IV). We were interested in episodes, during which the students in the group discussed aspects of the exercise (the program, the Java language, or the visualization), and we categorized these episodes according to the activities that accompany the conversation. We removed some of the categories (such as *executing code* and *working on solution*) that were irrelevant for the analysis since we chose to analyze episodes where students were, in fact, carrying out these activities. In addition, we added new categories related to different types of conversing (i.e., conversing with gesturing and conversing with drawing), and changed the priorities of the categories in order to prioritize discussions. Thus, if any kind of discussion happened, it was then assigned to one of the discussion categories. More specialized discussions were given a higher priority. We wanted to distinguish between discussions within the group and discussions with the teacher, so conversing to the instructor had higher priority so it could be distinguished from other types of discussions that happened within the group.

Each five-second period of the video was assigned to a single category based on the activities of the groups. Because the members of a group might perform several activities simultaneously, the priorities given for each category were used to resolve the ambiguities. An episode was assigned the category with the highest priority (lower numbers mean higher priority). Furthermore, we assigned the number of participants for each activity in order to determine if certain EET levels increase or reduce the participation of students.

Table IV. Activity Categories

Priority	Category	Description
1	1 Conversing to an instructor	The students discuss the exercise with the instructor.
2	2 Conversing with gesturing	The students perform gestures such as pointing at the screen when discussing the exercise.
3	3 Conversing with drawing	The students discuss the exercise with the help of drawing.
4	4 Conversing	The students discuss the exercise (without any of the above activities).
5	5 Listening to an instructor	The students are listening to the instructor who is talking with the group or announcing something to the whole class.
5	6 Looking at or searching for course materials, Internet resources or example.	Self-explanatory.
5	7 Reviewing the exercise	The students are looking at the online or hard-copy description of the problem.
5	8 Reviewing error messages	The students are reviewing error messages produced by the environment.
6	9 Silent work	The students are working silently; for example, looking at the visualization or entering input without comment.
8	10 Other	No observable activity or the activity is off-task.
9	-1 Indeterminable	The event cannot be categorized, for example, because of a technical problem in the recording.

4.6.2 *Discussion Contents.* We wished to investigate the actual content of the students' discussions in order to determine if the engagement level had any impact on the contents. If an activity was classified as (i) any type of conversing, (ii) other but it contained talking, or (iii) undetermined but it contained talking, it was further classified into one of ten discussion content categories (see Table V). The categories were exclusive, and the decision to classify an event into a category was based upon the discussions during the five-second period. If several categories were applicable, we used the category with the highest priority, and if there were several with the same priority then the one that happened first was chosen. The categories in Table V were adapted from Hundhausen and Brown [2008]. As that study dealt with work at the algorithmic level, we changed the relevant content categories so that they refer to programs instead of algorithms. Furthermore, we removed categories that were related to the ALVIS animation system that we did not use, and we added a category, *programming concepts*.

4.6.3 *Transactive Reasoning.* Since transactive reasoning has been found to have a positive impact on learning outcomes [Teasley 1997], we measured the amount of transactive reasoning in order to see if it can help determine how visualization affects the collaboration process.

If an activity was classified as any type of conversing, as other but it contained talking, or as undetermined but it contained talking, it was further clas-

Table V. Discussion Content Categories

Priority	Category	Description
1	1 Program	The content relates to the program code: "This line contains a while-loop."
1	2 Program behavior	The content relates to the program's behavior: "Now it repeats 10 times."
1	3 Programming concepts	The content relates to programming concepts in general not directly related to the current program: "What does `double` mean?"
1	4 Tool	The content relates to the tools currently being used tool: "How can I display the value of a variable?"
1	5 Error detection	The content relates to programming errors and their detection: "I spotted an error!"
1	6 Error correction	The content relates to the correction of an error: "If we change the value of this variable, that will solve the problem."
1	7 Visualization	The content relates to understanding the visualization itself: "How is this box related to the program?"
2	8 On-topic (other)	The content relates to the current task but cannot be placed into one of the previous categories.
2	9 Off-topic	Self-explanatory.
3	10 Indeterminable	The content cannot be categorized, for example, because of a technical problem in the recording.
0	-1 Not applicable	There is no content in this activity; for example, there is no talking in the segment.

sified into one of twelve transactive reasoning categories (see Table VI). These categories were exclusive. The decision to classify an event was based on the discussions and activities during a five-second block of the video. If several categories were applicable, we used the category with the highest priority, and if there were several categories with the same priority then the one that happened first was chosen. The categories in Table VI were adapted from Teasley [1997] and Berkowitz and Gibbs [1983]. We added the prediction category used by Teasley [1997] to the categories of Berkowitz and Gibbs [1983].

5. RESULTS

5.1 Inter-Rater Reliability

In order to test the reliability of the classification schemes used in the study, a set of ten episodes (a total of 240 five-second blocks) were classified by two raters, the first author, who classified all the video material used in the study, and the second author, who classified only the set of ten episodes.

In a pilot, both raters analyzed three other episodes in order to reach agreement on how to classify the observed behavior. The classification schemes and the coding manual were updated as a result of the discussions leading to consensus between the raters.

The inter-rater reliabilities are presented in Table VII. All Cohen's κ values indicated that it is very unlikely ($p < 0.001$) that this level of agreement is achieved by chance. Furthermore, the *EET* and Activities classification has a

Table VI. Transactive Reasoning Categories

Priority	Category	Description
1	0 Prediction	A student tries to predict what will happen next and justifies the prediction.
1	1 Feedback Request	A student ensures that others understand or agree with his/her position
1	2 Paraphrase	A student paraphrases a discourse of another student in order to demonstrate that he/she understands it.
1	3 Justification	A student justifies his/her position or reasoning.
1	4 Juxtaposition	A student explains the differences between the positions or reasoning of other students and his/her own.
1	5 Completion	A student completes another student's reasoning, for example, by filling out an unfinished sentence.
1	6 Clarification	A student explains his/her reasoning in order to ensure that others understand it.
1	7 Refinement	A student elaborates or qualifies his/her position in order to to defend against criticism.
1	8 Extension	A student elaborates on a previous discourse.
1	9 Criticism	A student criticizes the reasoning or position of another student and explains the reason for the criticism.
1	10 Integration	A student combines different views into one common statement.
2	11 No transactive reasoning	The discussion contains no transactive reasoning.
0	-1 Not applicable	This categorization is not applicable; for example, there is no talking in the segment.

Table VII. Inter-rater Reliabilities. * $p < 0.001$

Classification Scheme	Agreement Percentage	Cohen's κ
EET	76.3%	(0.66 *)
Activities	76.3%	(0.68 *)
Number of Participants	58.3%	(0.42 *)

substantial agreement and the Number of Participants has moderate agreement based on classification of Cohen's κ-measures given by Landis and Koch [1977].

For the two other categorizations (Discussion Contents and Transactive Reasoning) inter-rater reliabilities were low. We believe that this was due to several factors: 1) subtle cues of the discussion contents or transactive reasoning (e.g., only one word could indicate if students discussed about program, its behavior or programming concept) which might have been misinterpreted by the raters, and 2) noise in the natural environment (i.e., classroom with several groups) made it sometimes difficult to interpret exactly what the group was discussing. We still include in the article results of the discussion contents and transactive reasoning. Although they cannot be used as definitive evidence supporting our hypothesis, the results do indicate that these categorizations are consistent with the previous categorizations.

Table VIII. The Distribution of Activities on Different EET Levels

	Conversing w/I	Conversing w/G	Conversing	Listen t/I	Review Ex	Review EM	Silent	Count
No viewing	6.8%	12.4%	27.8%	27.2%	0.0%	1.2%	24.7%	162
Viewing	4.4%	6.1%	25.4%	15.7%	0.9%	0.0%	47.5%	343
Controlled viewing	8.6%	0.0%	22.9%	34.3%	0.0%	0.0%	34.3%	35
Entering input	5.5%	8.0%	42.9%	12.3%	0.0%	1.2%	30.1%	163
Changing	0.0%	0.0%	0.0%	100.0%	0.0%	0.0%	0.0%	2
Modifying	0.0%	16.7%	50.0%	0.0%	0.0%	0.0%	33.3%	6
Constructing	22.2%	0.0%	0.0%	77.8%	0.0%	0.0%	0.0%	9
Overall	5.6%	7.6%	29.6%	19.3%	0.4%	0.6%	36.9%	720

Legend: w/I = with instructor; w/G = with gestures; t/I = to instructor; Ex = exerices; EM=error messages

5.2 Activities

The distribution of the activities on each EET level is presented in Table VIII. EET levels that contained fewer than ten observations or that had categories that did not contain any observations (controlled viewing, changing, modifying and constructing) were excluded from the analysis. Activity columns that contained observations only on one or two EET levels (i.e., looking at or searching for examples or course materials, and reviewing the exercise) were also excluded due to the restrictions of χ^2-test. These categories only contributed less than eight percent of the overall data.

The distributions of the EET levels no viewing, viewing and entering input were compared, first collectively ($\chi^2(8) = 48.5, p < .01$), and then pairwise (no viewing vs. viewing $\chi^2(4) = 28.4, p < .01$; no viewing vs. entering input $\chi^2(4) = 17.0, p < .01$; viewing vs. entering input $\chi^2(4) = 20.9, p < .01$). All tests were found to be statistically significant, meaning that the EET level has an effect on the distribution.

Figure 3 shows the distributions of activities for the three most common EET levels collapsed into three columns. The different forms of conversing are combined into one, *sum of conversing*, and the categories that did not have observations for all engagement levels were removed as they contributed less than eight percent of the data. The distributions were first compared collectively ($\chi^2(4) = 41.6, p < .01$), and then pairwise (no viewing vs. viewing $\chi^2(2) = 25.0, p < .01$; no viewing vs. entering input $\chi^2(2) = 37.6, p < .01$; viewing vs. entering input $\chi^2(2) = 28.2, p < .01$). All tests were found to be significant. Figure 3 shows that entering input produced the greatest amount of conversation. When students were not viewing a visualization, they listened to the teacher more often than on any other EET level. When students were viewing a visualization they were more often silent.

Figure 4 illustrates the difference between conversing and silent activities performed by the groups when either on viewing or on entering input level. On entering input level over half of the activities contained discussions whereas in viewing level the percentage was approximately 35%. Almost the opposite happens with the amount of silence.

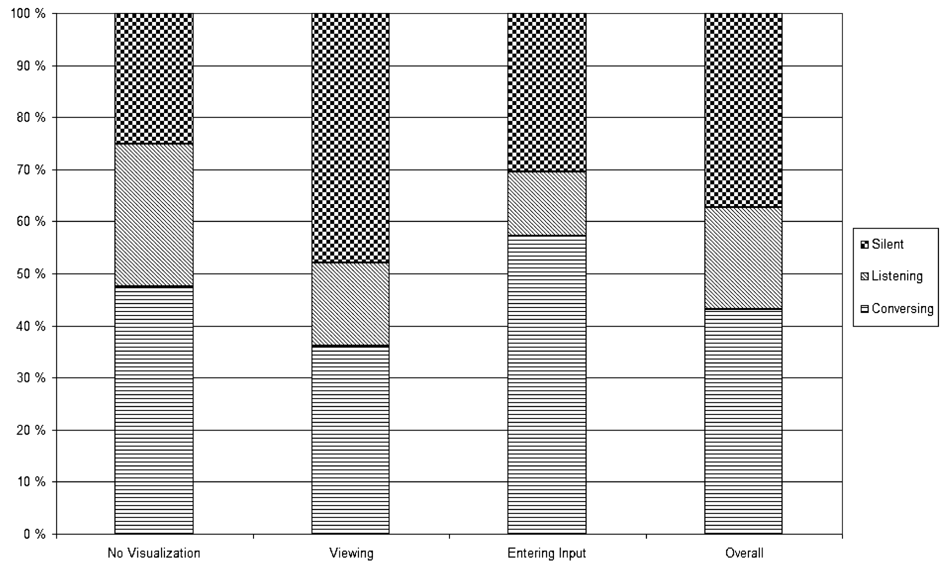


Fig. 3. Activity distributions on different EET levels.

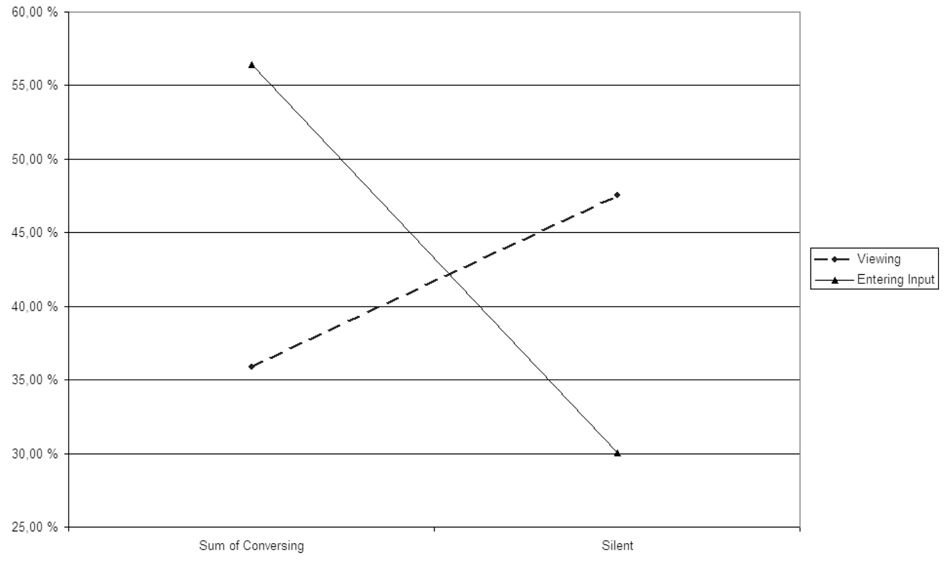


Fig. 4. Viewing and entering input EET-levels compared on sum of conversing and silent categories.

Table IX shows how the distribution of activities differs on each EET level, depending on the tool used by the students. Jeliot provides support only for the first four EET levels and the modifying level, which did not appear in the data, therefore, there are no results for the higher levels. The within-tool distributions of the activities for the most common EET levels (no viewing, viewing and entering input) were compared to each other and differences

Table IX. Distribution of Activities on Each EET Level and When Using a Particular Tool. (Legend as in Table VIII)

	Conversing w/I	Conversing w/G	Conversing	Listen t/I	Silent	Count
Jeliot						
No viewing	4.8%	16.7%	23.8%	33.3%	21.4%	42
Viewing	0.4%	4.9%	22.5%	12.7%	59.6%	245
Controlled viewing	8.3%	0.0%	41.7%	16.7%	33.3%	12
Entering input	1.8%	10.9%	25.5%	10.9%	50.9%	55
BlueJ						
No viewing	7.6%	11.0%	29.7%	25.4%	26.3%	118
Viewing	14.7%	9.5%	33.7%	24.2%	17.9%	95
Controlled viewing	8.7%	0.0%	13.0%	43.5%	34.8%	23
Entering input	7.6%	6.6%	52.8%	13.2%	19.8%	106
Changing	0.0%	0.0%	0.0%	100.0%	0.0%	2
Modifying	0.0%	16.7%	50.0%	0.0%	33.3%	6
Constructing	22.2%	0.0%	0.0%	77.8%	0.0%	9

Table X. The Average Number of Participants at Each EET Level

EET	Average Number of Participants
No viewing	3.1
Viewing	3.1
Controlled viewing	3.5
Entering input	2.9
Changing	4.0
Modifying	3.0
Constructing	3.2
Overall	3.1

were found to be statistically significant (Jeliot: $\chi^2(8) = 35.0, p < .01$; BlueJ: $\chi^2(8) = 19.9, p < .05$). The activity distributions on both tools are similar to the overall distribution shown in the Figure 3 (cf., previous paragraph).

5.3 Participation

Table X shows the average number of participants on each EET level. In the most frequently occurring levels (no viewing, viewing, entering input), the number of participants is almost the same. There were no statistically significant differences and the number of participants on each activity is large enough to argue that the students were truly collaborating.

5.4 Discussion Contents

All the activities that contained discussions were further classified based on their discussion contents. Table XI shows the distribution of the frequency of topics during the conversations when the students were working on one of the EET levels.

The variability of the discussion contents between EET levels was large; for example, on the no viewing level, the program category contains 25.7% of the data, while there are no data for program category on the *viewing* level.

Table XI. Discussion Topics Distribution on Each EET Level

EET / Content	P	PB	PC	Tool	ED	EC	Visual-ization	On-topic	Off-topic	I	Count
No viewing	25.7%	24.9%	0.0%	9.9%	4.0%	9.9%	0.0%	22.8%	2.0%	1.0%	101
Viewing	0.0%	33.1%	2.1%	14.8%	9.2%	2.8%	3.5%	29.6%	2.1%	2.8%	142
Controlled viewing	0.0%	9.5%	9.5%	42.9%	4.8%	0.0%	0.0%	33.3%	0.0%	0.0%	21
Entering input	2.1%	37.9%	4.2%	11.6%	4.2%	1.1%	1.1%	34.7%	0.0%	3.2%	95
Changing	0.0%	0.0%	0.0%	50.0%	0.0%	0.0%	0.0%	50.0%	0.0%	0.0%	2
Modifying	0.0%	25.0%	0.0%	0.0%	50.0%	25.0%	0.0%	0.0%	0.0%	0.0%	4
Constructing	33.3%	0.0%	33.3%	33.3%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	3

Legend: P=Program; PB=Program behavior; PC=Program concepts; ED=Error detection; EC=Error correction; I=Indeterminable

Table XII. Transactive Reasoning on Each EET Level

EET	Transactive reasoning	Count
No viewing	5.0%	101
Viewing	3.6%	139
Controlled viewing	0.0%	21
Entering input	6.5%	93
Changing	0.0%	2
Modifying	33.3%	3
Constructing	0.0%	3

Therefore, the χ^2 test might not be reliable. The three most frequent EET levels (no viewing, viewing and entering input) were included into the analysis. The test was first done for all three levels showing that the discussion content distributions and the EET levels are related ($\chi^2(18) = 85.1, p < .01$). However, pairwise comparison revealed that only differences between no viewing and viewing ($\chi^2(9) = 54.1, p < .01$), and between no viewing and entering input ($\chi^2(9) = 39.6, p < .01$) were significant, meaning that those levels have different discussion content distributions. This also means that the distributions of the discussion contents were very similar on both of the EET levels, viewing and entering input, based on the statistical tests.

5.5 Transactive Reasoning

Less than five percent of all the discussions contained transactive reasoning. Therefore, we do not report the results for each type separately, but, rather, in Table XII we show the percentages of discussions that contained and did not contain transactive reasoning. From the most frequent EET levels, entering input had the highest percentage of transactive reasoning; however, the differences were not statistically significant.

6. DISCUSSION

The study partially confirms our hypothesis that the level of engagement on which students select to work with the visualization tool affects the quality of collaboration; that is, the engagement level and the interaction between the students are correlated. This seems to be true especially when a tool supports engagement on the levels viewing and entering input: the latter increases the amount of discussion significantly and reduces the time when students are silent, the former does the opposite. The level entering input also increases the amount of transactive reasoning, although the differences are not statistically significant. The increase in the amount of discussion changes neither the contents of the discussions nor the participation of the students, and similar numbers of students discuss similar topics on both levels, viewing and entering input.

To summarize, this study shows that EET levels are positively correlated with the amount of interaction, and it is interaction which is an important component of the successful collaboration (Section 2.4). However, this does not change the contents of the interactions and actually increases, although not significantly, the amount of transactive reasoning, which is positively correlated with learning outcomes.

At the lowest level of engagement (i.e., no viewing), the hypothesis does not hold and the quality of the collaboration lies between that of entering input and viewing. It seems that the no viewing level promotes different kinds of interactions and communication among the students than the other levels. Students' discussions are more about the program code, which is natural, because the program code is often the only representation available at this level. Furthermore, students listen to and discuss with the instructor more often, which can mean either that they are seeking help or that they are listening to instructor's guidance more often when they are working on this level or that the teacher guides them to this level when he/she comes to help the students. It is reasonable to assume that this compensates for the lack of help and guidance that is available from the visualization tool. This result partially replicates the results of Hundhausen and Brown [2008] discussed in the Section 2.3.

The average number of participants taking part in the collaboration is roughly the same at all levels of engagement. Three students seem to be optimal for collaborating on a single computer that is running a visualization tool. We observed from the videos that in many cases when there were four students, one of them sat far from the computer and was passive. Nevertheless, for some groups the amount of collaboration among all four students was high, a result we attribute to their interpersonal skills.

Although a comparison of Jeliot and BlueJ is not within the scope of this article, a preliminary analysis indicates that it was the level of engagement that was more influential than the specific tool. A currently unpublished follow-up analysis of video protocols [Korhonen et al. 2008] from a study, in which students were using the TRAKLA2 system to learn data structures in small groups (see the report of the original study in Myller et al. [2007b] and in Laakso et al. [2008] on students' learning results), found the same correlation between engagement levels and the amount of communication and collaboration. This also supports the hypothesis and shows that it is independent of the tool that is being used.

Our experiment was carried out in an actual classroom and it has high ecological validity, but this also means that we could not control all variables as we could have done in an experimental setting. Therefore, we could not establish that the engagement level caused the increase in the interactions and the quality of collaboration. We can only say that they are correlated, so the causation, if any, could be in either direction. On the one hand, there are certain levels of engagement that were not always controlled by the students, but happened as a side effect of the animation or other actions. For example, when students were viewing an animation, students needed to enter input whenever the visualized program needs input. Thus, they engaged on entering input, although they only selected the viewing of the animation. On the other hand, an activity could change the engagement level; for example, when students were discussing with the instructor, s/he proposed the use of a visualization on a certain engagement level (e.g., seeing the source code instead of the animation).

In this study, we have not taken into consideration how the correlation between EET levels and collaboration and communication develops over time. Thus, based on the current research, we can definitely describe the correlation only for the first week of the first course on programming. In future research, we plan to study the effects of the use of visualizations on collaboration by analyzing the video materials from several weeks of the course.

6.1 Threats to Validity and Reliability

Although we have tried to make sure that all the steps of the research process would maintain the internal validity and reliability, there are issues that might have affected the results. We have aimed to have a high ecological validity, so we carried out the study in a real classroom environment. We could not randomize the students into the different sessions; however, we randomized the students within the sessions into small groups and by using a post-hoc analysis, we checked that the background variables of the students were similar between the sessions. Thus, we expect that the collected data from several groups is comparable and aggregable.

When the video data was sampled, we selected episodes in the materials from one week. There could be bias in the random selection of the episodes that could affect the results. We tried to minimize the bias by selecting a similar number of episodes from each group for each tool. This should have balanced out the influence of a single group to the final results.

The videos were coded using different coding schemes. The coding schemes were checked for reliability by comparing the results of two coders on a subset of the data. It was found that most of the coding schemes are reliable except for the discussion contents and transactive reasoning categories. Thus, we use them as secondary evidence not as the primary evidence for our hypothesis. The reliability of this secondary evidence needs to be further tested in the future studies.

Some of the EET levels were not completely assessed because we did not have enough data from those levels. Given the high inter-rater reliabilities in classifying all the EET levels, we believe that additional data collection will enable us to analyze other levels in the future experiments.

7. CONCLUSION

We have presented an extension to the Engagement Taxonomy, the Extended Engagement Taxonomy, and used it to investigate the mutual relationship between visualization tools and collaborative learning. Our empirical study demonstrated support for the hypothesis that increasing the level of engagement between learners and the visualization tool results in a higher positive impact of the visualization on the collaboration process.

The EET can be used during the design and development of visualization tools for collaborative learning. There are several design implications that are already implemented in practice:

—We have added a capability for automatic question generation to Jeliot 3 in order to support the EET level responding [Myller 2007].

—We believe that closer linking of BlueJ and Jeliot [Myller et al. 2007a] can increase the engagement level of students. Students should be able to see both BlueJ's object bench and Jeliot's animation at the same time, so that any modification of an object results in a change in the animation. This would allow a step-by-step construction of dynamic visualizations by the students.

—We have combined Jeliot 3 with Woven Stories 2, a collaborative authoring tool [Myller and Nuutinen 2006], in order to provide Jeliot 3 with collaborative editing support and augmenting visualizations that can support online collaborative learning.

There are also pedagogical implications of the findings:

—Higher levels of engagement provide more support for collaborative activities, so instructors should find ways to use visualization tools at these levels.

—The viewing level seems to reduce collaboration significantly because students become passive. Thus, engagement at this level should be avoided, if possible, or it should combined with explanations by the teacher, as was done by Ben-Bassat Levy et al. [2003] with positive results.

—The lowest level of engagement (no viewing) does not decrease interaction and collaboration of students as much as viewing. However, when this level of the EET is used in teaching, the instructor should be aware of the change in the focus of the discussions (the program code) and of the need to provide students with more help and guidance.

Our study suggests new directions in research on engagement in collaborative learning with visualizations. The first step should be an expansion of the analysis of the differences at a finer level of detail; for example, the contents of the discussions between the students could be further analyzed to determine what communicative resources they are referring to during the discussions (e.g., the visualization or the source code). Also, the interaction of learning and time-on-task should be analyzed in order to better understand how the role of the visualization during the students' learning process changes in long run.

Therefore, a longitudinal analysis of the collected data from this study (video materials from sessions during several weeks) could be used to analyze the effects of time and learning, for example, by using a time-series analysis.

We also plan to evaluate the question generation support added to Jeliot 3 and the closer linking of BlueJ and Jeliot. This should shed more light on the effects of the higher levels of the EET.

We think that it is important to create visualization tools that support several engagement levels—especially the higher ones—and make the instructors aware of the potentials that the higher engagement levels can provide so that students and their instructors can use the tools to their benefit during the collaborative learning.

ACKNOWLEDGMENTS

The authors would like to thank Andrés Moreno for the help during the data collection process.

REFERENCES

BEN-BASSAT LEVY, R., BEN-ARI, M., AND URONEN, P. A. 2003. The Jeliot 2000 program animation system. *Comput. Ed. 40*, 1, 15–21.

BERKOWITZ, M. W. AND GIBBS, J. C. 1983. Measuring the development of features in moral discussion. *Merill-Palmer Quar. 29*, 399–410.

BRYANT, S., ROMERO, P., AND DU BOULAY, B. 2005. Pair programming and the reappropriation of individual tools for collaborative programming. In *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work (SIGGROUP'05)*, M. Pendergast, K. Schmidt, G. Mark, and M. Ackerman Eds. ACM Press, 332–333.

EBEL, G. AND BEN-ARI, M. 2006. Affective effects of program visualization. In *Proceedings of the 2nd International Computing Education Research Workshop (ICER'06)*. ACM Press, 1-5.

EVANS, C. AND GIBBONS, N. J. 2007. The interactivity effect in multimedia learning. *Comput. Ed. 49*, 4, 1147–1160.

GALL, M. D., GALL, J. P., AND BORG, W. R. 2006. *Educational Research: An Introduction 8th Ed.* Allyn & Bacon.

GREEN, T. R. G. AND PETRE, M. 1996. Usability analysis of visual programming environments: A "cognitive dimensions" framework. *J. Vis. Lang. Comput. 7*, 131–174.

GRISSOM, S., MCNALLY, M., AND NAPS, T. L. 2003. Algorithm visualization in CS education: Comparing levels of student engagement. In *Proceedings of the 1st ACM Symposium on Software Visualization (SOFTVIS'03)*. ACM Press, 87–94.

HAASTER, K. V. AND HAGAN, D. 2004. Teaching and learning with BlueJ: An evaluation of a pedagogical tool. In *Proceedings of Informing Science + IT Education Conference (InSITE'04)*. Informing Science Institute, 455–470.

HÜBSCHER-YOUNGER, T. AND NARAYANAN, N. H. 2003. Constructive and collaborative learning of algorithms. *SIGCSE Bull. 35*, 1, 6–10.

HUNDHAUSEN, C. D. 2002. Integrating algorithm visualization technology into an undergraduate algorithms course: Ethnographic studies of a social constructivist approach. *Comput. Ed. 39*, 3, 237–260.

HUNDHAUSEN, C. D. 2005. Using end-user visualization environments to mediate conversations: A "communicative dimensions" framework. *J. Vis. Lang. Comput. 16*, 3, 153–185.

HUNDHAUSEN, C. D. AND BROWN, J. L. 2005. Personalizing and discussing algorithms within CS1 studio experiences: An observational study. In *Proceedings of the International Workshop on Computing Education Research (ICER'05)*. ACM Press, 45–56.

HUNDHAUSEN, C. D. AND BROWN, J. L. 2007. What you see is what you code: A "live" algorithm development and visualization environment for novice learners. *J. Vis. Lang. Comput. 18*, 1, 22–47.

HUNDHAUSEN, C. D. AND BROWN, J. L. 2008. Designing, visualizing, and discussing algorithms within a CS 1 studio experience: An empirical study. *Comput. Ed. 50*, 1, 301–326.

HUNDHAUSEN, C. D., DOUGLAS, S. A., AND STASKO, J. T. 2002. A meta-study of algorithm visualization effectiveness. *J. Vis. Lang. Comput. 13*, 3, 259–290.

JANSSEN, J., ERKENS, G., KANSELAAR, G., AND JASPERS, J. 2007. Visualization of participation: Does it contribute to successful computer-supported collaborative learning? *Comput. Ed. 49*, 4, 1037–1065.

JEHNG, J.-C. J. AND CHAN, T.-W. 1998. Designing computer support for collaborative visual learning in the domain of computer programming. *Comput. Hum. Behav. 14*, 3, 429–448.

KÖLLING, M., QUIG, B., PATTERSON, A., AND ROSENBERG, J. 2003. The BlueJ system and its pedagogy. *Comput. Science Ed. 13*, 4, 249–268.

KORHONEN, A., LAAKSO, M., AND MYLLER, N. 2008. How does algorithm visualization affect collaboration? Video analysis of engagement and discussions. *5th International Conference on Web Information Systems and Technologies (WEBIST'09)*. Submitted.

LAAKSO, M.-J., MYLLER, N., AND KORHONEN, A. 2008. Analyzing the extended engagement taxonomy in collaborative algorithm visualization. *J. Ed. Technol. Soc*. To appear.

LANDIS, J. R. AND KOCH, G. G. 1977. The measurement of observer agreement for categorical data. *Biometrics 33*, 159–174.

MCDOWELL, C., WERNER, L., BULLOCK, H. E., AND FERNALD, J. 2006. Pair programming improves student retention, confidence, and program quality. *Comm. ACM 49*, 8, 90–95.

MEIER, A., SPADA, H., AND RUMMEL, N. 2007. A rating scheme for assessing the quality of computer-supported collaboration processes. *International J. Comput. Support. Collab. Learn. 2*, 1, 63–86.

MORENO, A., MYLLER, N., SUTINEN, E., AND BEN-ARI, M. 2004. Visualizing program with Jeliot 3. In *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI'04)*. ACM Press, 373–380.

MYLLER, N. 2007. Automatic generation of prediction questions during program visualization. *Electron. Notes Theor. Comput. Sci. 178*, 43–49.

MYLLER, N., BEDNARIK, R., AND MORENO, A. 2007a. Integrating dynamic program visualization into BlueJ: The Jeliot 3 extension. In *Proceedings of the 7th IEEE International Conference on Advanced Learning Technologies*, J. M. Spector, D. G. Sampson, T. Okamoto, Kinshuk, S. A. Cerri, M. Ueno, and A. Kashihara Eds. IEEE Computer Society, 505–506.

MYLLER, N., LAAKSO, M., AND KORHONEN, A. 2007b. Analyzing engagement taxonomy in collaborative algorithm visualization. In *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE'07)*, J. Hughes, D. R. Peiris, and P. T. Tymann Eds. ACM Press, 251–255.

MYLLER, N. AND NUUTINEN, J. 2006. JeCo: Combining program visualization and story weaving. *Informatics Ed. 5*, 2, 267–276.

NAGAPPAN, N., WILLIAMS, L., FERZLI, M., WIEBE, E., YANG, K., MILLER, C., AND BALIK, S. 2003. Improving the CS1 experience with pair programming. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'03)*. ACM Press, 359–362.

NAPS, T. L. 2005. Jhavé – Addressing the need to support algorithm visualization with tools for active engagement. *IEEE Comput.- Graph. Appl. 25*, 5, 49–55.

NAPS, T. L. AND GRISSOM, S. 2002. The effective use of quicksort visualizations in the classroom. *J. Comput. Sci. Coll 18*, 1, 88–96.

NAPS, T. L., RÖSSLING, G., ALMSTRUM, V., DANN, W., FLEISCHER, R., HUNDHAUSEN, C., KORHONEN, A., MALMI, L., MCNALLY, M., RODGER, S., AND VELÁZQUEZ-ITURBIDE, J. Á. 2002. Exploring the role of visualization and engagement in computer science education. In *ITiCSE on Innovation and Technology in Computer Science Education (ITiCSE'02)*. (Working Groups Report). ACM Press, 131–152.

OECHSLE, R. AND MORTH, T. 2007. Peer review of animations developed by students. *Electron. Notes Theor. Comput. Sci. 178*, 181–186.

RAGONIS, N. AND BEN-ARI, M. 2005. On understanding the statics and dynamics of object-oriented programs. *SIGCSE Bull. 37*, 1, 226–230.

ROSCHELLE, J. 1996. Designing for cognitive communication: Epistemic fidelity or mediating collaborative inquiry. In *Computers, Communication & Mental Models*, D. L. Day and D. K. Kovacs Eds. Taylor & Francis, London, UK. 13–25.

RÖßLING, G. AND NAPS, T. L. 2002. A testbed for pedagogical requirements in algorithm visualizations. In *Proceedings of the Innovation and Technology in Computer Science Education (ITiCSE'02)*. ACM Press, 96–100.

SCAIFE, M. AND ROGERS, Y. 1996. External cognition: how do graphical representations work? *Int. J. Hum.-Comput. Stud. 45*, 2, 185–213.

SIMON, B., ANDERSON, R., HOYER, C., AND SU, J. 2004. Preliminary experiences with a tablet PC based system to support active learning in computer science courses. In *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE'04)*. ACM, 213–217.

SPADA, H., MEIER, A., RUMMEL, N., AND HAUSER, S. 2005. A new method to assess the quality of collaborative process in CSCL. In *Computer Supported Collaborative Learning 2005: The Next 10 Years*, T. Koschmann, D. Suthers, and T. W. Chan Eds. Lawrence Erlbaum, Mahwah, NJ. 622–631.

SUTHERS, D. D. AND HUNDHAUSEN, C. D. 2003. An experimental study of the effects of representational guidance on collaborative learning processes. *J. Learn. Sciences 12*, 2, 183–219.

SUTHERS, D. D., HUNDHAUSEN, C. D., AND GIRARDEAU, L. E. 2003. Comparing the roles of representations in face-to-face and online computer supported collaborative learning. *Comput. Ed. 41*, 4, 335–351.

TEASLEY, S. 1997. Talking about reasoning: How important is the peer in peer collaboration. In *Discourse, Tools and Reasoning: Essays on Situated Cognition*, L. Resnick, R. Säljö, C. Pontecorvo, and B. Burge Eds. Springer, Berlin, Germany. 361–384.

WILLIAMS, L., KESSLER, R. R., CUNNINGHAM, W., AND JEFFRIES, R. 2000. Strengthening the case for pair programming. *IEEE Softw. 17*, 4, 19–25.

Received January 2008; revised September 2008; accepted November 2008

P2.

2

Myller, N., Laakso, M., and Korhonen, A. (2007). Analyzing engagement taxonomy in collaborative algorithm visualization. In Hughes, J., Peiris, D. R., and Tymann, P. T., editors, *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE ’07)*, pages 251–255, New York, NY, USA. ACM Press.

Analyzing Engagement Taxonomy In Collaborative Algorithm Visualization

Niko Myller
Department of Computer Science and Statistics
University of Joensuu
P.O. Box 111
FI-80101 Joensuu
Joensuu, Finland
nmyller@cs.joensuu.fi

Mikko Laakso
Department of Information Technology
University of Turku
22014 Turun Yliopisto
Turku, Finland
milaak@utu.fi

Ari Korhonen
Department of Computer Science and Engineering
Helsinki University of Technology
P.O. Box 5400
FI-02015 TKK
Espoo, Finland
archie@cs.hut.fi

ABSTRACT

More collaborative use of visualizations is taking place in the classrooms due to the introduction of pair programming and collaborative learning as teaching and learning methods. This introduces new challenges to the visualization tools, and thus, research and theory to support the development of collaborative visualization tools is needed. We present an empirical study in which the learning outcomes of students were compared when students were learning in collaboration and using materials which contained visualizations on different engagement levels. Results indicate that the level of engagement has an effect on students' learning results although the difference is not statistically significant. Especially, students without previous knowledge seem to gain more from using visualizations on higher engagement level.

Categories and Subject Descriptors

K.3.2 [**Computer Science Education**]: Computer & Information Science Education—*Computer Science Education*

General Terms

Algorithms, Human Factors

Keywords

Algorithm Visualization, Algorithm Simulation, Collaborative Learning, Engagement Taxonomy

1. INTRODUCTION

Since its introduction, Algorithm Visualization (AV) has been hoped to solve problems related the learning of data structures and algorithms. However, empirical evaluations have yielded mixed results when determining the usefulness of visualizations as teaching and learning aids over traditional methods [8]. Thus, researchers have sought explanations for the mixed results as well as better grounds to justify the use of visualizations in teaching. In a meta-analysis of the research on AV, Hundhausen et al. [8] concluded that the activities performed by the students are more important than the content of the visualization. This led to the analysis of different engagement levels by ITiCSE working group [12] that proposed *Engagement Taxonomy* (ET) to describe the various types of activities that students perform with visualizations and their effect on learning.

Collaboration have become accepted and popular in Computer Science education. A good example is the benefits of pair programming [10, 16]. Thus, visualizations are also used in collaborative learning. Collaboration introduces new challenges for the visualization tools. For example, the exchange of experiences and ideas and coordination of the joint work are needed, when students are not working individually [15]. Furthermore, visualizations can provide a shared external memory that can initiate negotiations of meanings and act as a reference point when ideas are explained or misunderstandings are resolved [15]. This implies that also theory is needed to guide the development and research of the visualization tools for collaborative learning.

Currently, the applicability of ET framework in collaborative use of visualizations has not been researched. We wanted to test the impact of ET level of the visualization to the performance differences when visualizations are used in collaboration. We present an empirical study, in which learning materials containing visualizations on different ET levels were compared when students were collaboratively learning concepts related to binary heap. Although statistically significant differences were not detected, the results indicate that the engagement level of the visualizations has an effect on the performance when students are working in pairs.

2. PREVIOUS RESEARCH

As an attempt to describe the mixed results of previous research in AV usage (cf. [8]) in learning and teaching of algorithms and data structures, Engagement Taxonomy (ET) was introduced by Naps et al. [12]. The central idea of the taxonomy is that the higher the engagement between learner and the visualization, the higher the positive effect on learning outcomes are. ET consists of six levels of engagement between the user and the visualization:

No viewing – There is no visualization to be viewed.

Viewing – The visualization is only looked at.

Responding – Visualization is accompanied with questions which are related to the content of the visualization.

Changing – Modification of the visualization is allowed, for example, by varying the input data set or algorithm simulation.

Constructing – Visualization of program or algorithm is created.

Presenting – Visualizations are presented to others for feedback and discussions.

ET has been used in development of AV tools and several studies have utilized the framework and provided further support for it (e.g [4, 11]). Other studies have provided similar support without actually utilizing the ET framework in the design of the study [1]. In addition to this, research in educational psychology and multimedia learning has also got similar results [3].

Although several AV tools have been developed and empirical studies carried out, the collaborative use of AV tools is not well researched. To our knowledge there have not been any studies that would have tested the applicability of ET in collaborative learning with visualizations.

There are a few studies that have research some aspects of collaborative use of visualizations in learning. The work of Hundhausen et al. [6] studied the collaborative aspects of AV construction and presentation. This work led into the development of a visualization tool, ALVIS, which supports construction and presentation of AVs in small groups [7]. Their results seem to indicate that ET is applicable in the context of collaborative learning. Hübscher-Younger and Narayanan [5] developed a web-based system that allowed students to post their own algorithm representations (e.g. text, pictures, animations or multimedia) and discuss them on the web. The research concluded that students who actively participated in this activity achieved higher grades than the passive students who might have only viewed and commented others' presentations.

From a more general perspective, there are studies of analyzing the use of visualizations in collaboration. For instance, Suthers and Hundhausen [15] have performed research in the area of scientific inquiry. They compared the effects of different representations (e.g. matrix, graph or text) when students were collecting and analyzing data, hypotheses and their evidential relations. Their research showed that the form of the visualization and what kinds of interactions it drives have an effect on the collaboration process.

Roschelle [13] studied pairs of students using the learning environment of Newtonian physics and analyzed their learning outcomes as well as the process that led to those outcomes. It was recognized that learning tools and especially visualizations used in collaboration should focus more on supporting communication rather than presenting the underlying model as accurately as possible. The last lesson in the paper is, "one should design activities which actively engage students in doing and encounter meaningful experiential feedback as a consequence of their actions". The analysis of the interaction between external presentation and its users has been identified as a key research area by Scaife and Rogers [14]. These findings are supporting the applicability of ET in the context of collaborative learning.

3. TRAKLA2 OVERVIEW

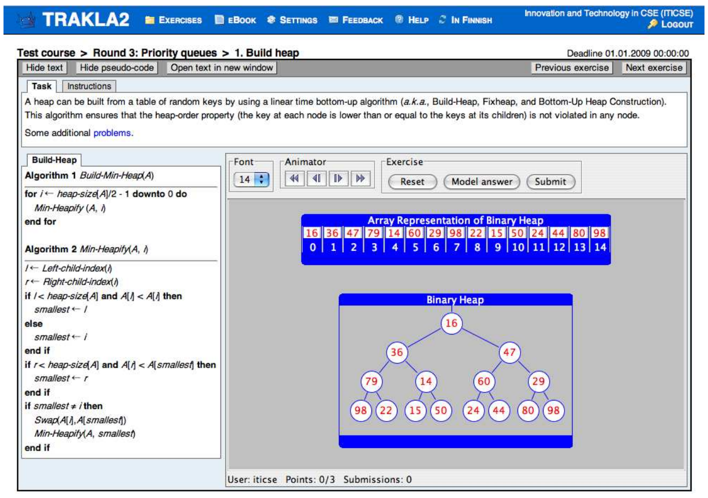
TRAKLA2 is a practicing environment for *visual algorithm simulation exercises* [9] that can be assessed automatically. The system distributes individually tailored tracing exercises to students and provides automatically feedback about students' solutions. In visual algorithm simulation exercises, a student directly manipulates the visual representation of the underlying data structures. Thus, the student manipulates real data structures through GUI operations with the purpose of performing the same changes on the data structures that the actual algorithm would do. An answer to an exercise is a sequence of discrete states of the data structures, and the task is to perform the correct operations that will cause the transitions between each of the two consecutive states.

Each TRAKLA2 exercise page consists of a description of the exercise with links to other pages that introduce the theory and examples of the algorithm in question, instructions how to interact with the GUI, code window, and an interactive Java applet. The current exercise set consists of over 40 assignments on basic data structures, sorting, searching, hashing, and graph algorithms.

Let us consider the exercise in Figure 1. The student is supposed to manipulate the visual representation(s) of the Binary Heap data structure by invoking context-sensitive *drag-and-drop operations*. The idea is to simulate the linear time BuildHeap algorithm. The manipulation can be done in either of the representations shown in the figure (array representation or binary tree representation). A key can be sifted up in terms of *swap operations* with its parent until the heap property is satisfied (the key at each node is smaller than or equal to the keys of its children). A single swap operation is performed by dragging and dropping a key in the heap on top of another key.

An exercise applet is initialized with *randomized input data*. The BuildHeap exercise, for example, is initialized with 15 numeric keys that correspond to the priority values. The student can *reset the exercise* by pressing the *Reset* button at any time. As a result, the exercise is reinitialized with new random keys. When attempting to solve the exercise, the student can *review the answer* step by step using the *Animator* panel. Moreover, the student can *Submit* the answer in which case the answer is assessed and immediate feedback is delivered. The feedback reports the number of correct steps out of the total number of steps in the exercise.

An exercise can be submitted unlimited times. However, a solution for a single instance of an exercise with certain input data can be submitted only once. In order to resubmit a solution to the exercise, the student has to reset the exercise and start over with new randomized input data. A student can also review a *Model answer* for each attempt. It is represented in a separate window as an algorithm animation accompanied with pseudo code animation so that the execution of the algorithm is visualized step by step. The states of the model solution can be browsed back and forth



Test course > Round 3: Priority queues > 1. Build heap

Deadline 01.01.2009 00:00:00

A heap can be built from a table of random keys by using a linear time bottom-up algorithm (*a.k.a.*, Build-Heap, Fixheap, and Bottom-Up Heap Construction). This algorithm ensures that the heap-order property (the key at each node is lower than or equal to the keys at its children) is not violated in any node.

Some additional problems.

```
Algorithm 1 Build-Min-Heap(A)
for i ← heap-size[A]/2 - 1 downto 0 do
  Min-Heapify (A, i)
end for

Algorithm 2 Min-Heapify(A, i)
l ← Left-child-index(i)
r ← Right-child-index(i)
if l < heap-size[A] and A[l] < A[i] then
  smallest ← l
else
  smallest ← i
end if
if r < heap-size[A] and A[r] < A[smallest] then
  smallest ← r
end if
if smallest ≠ i then
  Swap(A[i],A[smallest])
  Min-Heapify(A, smallest)
end if
```

User: iticse Points: 0/3 Submissions: 0

Figure 1: TRAKLA2 exercise page.

using similar animator panel as in the exercise. For obvious reasons — after opening the model solution — the student cannot submit a solution until the exercise has been reset and resolved with new random data.

4. EXPERIMENT

To summarize the previous sections, the collaborative use of AV tools have not been researched, yet the need for this kind of research emerges from the increasing use of visualization tools in collaborative learning. In addition, we hypothesize that ET framework can be used to predict performance differences when visualizations are used in collaboration. Previous research supports this view, although it has not be directly tested before.

In order to test our hypothesis, we carried out an experiment in which we compared learning outcomes of students who were collaboratively using visualizations which were on different ET level, namely *viewing* and *changing*. Participants were university students on a data structures and algorithms course at the University of Turku. We utilized TRAKLA2[9] in order to provide students with algorithm simulation exercises that act on the ET level *changing*. However, the students did not have the option to reset the exercise with new input data, but they had to work with a single exercise during the whole session. The animations that students in *viewing* condition used were similar in nature to the model answers provided by TRAKLA2 system.

4.1 Method

The study was between subject design with pre-test and post-test. We had one between subject factor: the ET level of the visualizations in the learning materials, namely *viewing* or *changing*. The unit of analysis is a student even though the learning was done in pairs.

The learning materials contained textual materials that were same for both conditions. In the *changing* condition textual materials were accompanied with TRAKLA2 [9] algorithm simulation exercises related to the binary heap. Pairs in the *viewing* condition were presented with animations about the operations of the binary heap that were similar to TRAKLA2 exercises. In addition, pairs in both conditions were given exercise that asked questions on binary heap that were supposed to be answered during the learning process. In this way, we tried to motivate the learning and make sure that the possible differences are due to controlled variable (level of engagement), and not because pairs in one condition performed cognitively more demanding activities or used more time on the tasks [4, 8].

Quantitative results were analyzed with two-tailed t-test, and Bonferroni correction was applied when more than one related t-test was performed.

4.2 Subjects

Experiment was performed on a course of data structures and algorithms at the University of Turku, Finland. Students were mainly second year students, however some students from other years were also on the course. Students were randomized to the computer lab sessions and sessions were randomly assigned to each condition with the limitation that parallel sessions belonged to different conditions. The total number of participating students was 105 ($n = 105$) divided into 6 groups (3 control groups having *viewing* condition and 3 treatment groups having *changing* condition). There were 52 students in the treatment group and 53 students in the control group.

The most of the students were studying in the Faculty of Mathematics and Natural Sciences. There were 93 of 105 students who had Computer Science as the major and ten students had major in mathematics. There were also two students from other departments, one from the Faculty of Social Sciences and one from The Faculty of Humanities. Most of the students were from first (17), second (36) or third (27) year which sums up to 80 students out of 105. Furthermore, there were 12 fourth year students and 13 students who had started their studies in 2002 or earlier.

4.3 Materials

The pre-test and post-test consisted of six questions which were exactly the same. The first question was asking if a given array is a heap. In the second question, we asked if an ordered array is a heap or not. The third and the fourth question asked the students to write an answer where is the place of the smallest value in the minimum binary heap, or in the maximum binary heap, respectively. In the fifth question, students where asked to write down a given binary heap's heap property. The last question asked the students to draw a binary tree representation from given array presentation of a minimum binary heap.

4.4 Procedure

Study was performed on the second week of the course at the computer lab sessions that lasted for 2 hours. There were a total of 6 sessions and they were run on a single day, so that there were two sessions running at the same time, one for each condition.

In the beginning of the session, students could freely form pairs with their peers. If there were an odd number of students one group consisted of 3 students. Each pair was allocated to a single computer. Students gave their consent to participate to the experiment and took an individual pre-test in which they needed to answer questions related to binary heaps in 15 minutes.

After pre-test, students were presented with the learning materials of that condition. Students had 45 minutes to

Table 1: The means of the differences between pretest and posttest scores for each question and for the total points between the conditions. The final row shows the averaged normalized difference in in total scores between pre- and post-test calculated for each student with formula $(p_{posttest} - p_{pretest})/(24 - p_{pretest})$. Standard deviations are in the parenthesis.

Question	Viewing	Changing
1	1.62 (2.41)	2.02 (2.55)
2	2.10 (2.06)	2.15 (2.24)
3	2.25 (2.14)	2.62 (1.90)
4	1.77 (1.38)	2.36 (1.11)
5	2.46 (1.58)	2.74 (1.50)
6	3.06 (1.66)	3.06 (1.68)
Total	13.25 (6.90)	14.94 (7.05)
Normalized Total	0.66 (0.29)	0.73 (0.27)

go through the learning materials. Students learning was monitored by recording their talking and capturing their computer screens. The analysis of this data is out of the scope of this paper.

The session ended with an individual post-test that tested same questions as pre-test. In addition to this, post-test contained more demanding questions in order to see, if differences between conditions can be detected in them. However, the results of these additional questions are not analyzed in this paper. Students were given 30 minutes to answer the questions in the post-test.

Each question in the pre- and post-tests was analyzed in a scale from 0 and 4. Zero points meant that there was less than 25 percent correct in the answer, and each point meant 25 percent increase in the correctness of the answer.

4.5 Results

There was no significant difference in the pre-test scores between students in the each condition. Average scores (standard deviations are in the parenthesis) in pre-test for *viewing* and *changing* conditions were 4.65 (5.81) and 4.34 (5.41), respectively . As expected the post-test scores were significantly higher than the pre-test scores for both groups (viewing: $t_{pw}(51) = -13.84, p < .001$, changing: $t_{pw}(52) = -15.44, p < .001$).

To analyze the differences of the learning outcomes between the conditions, we subtracted the pre-test score from each student's post-test score. The average increases for each question and condition are presented in Table 1. Only the difference in question 4 was found to be under the α=0.05 ($t(103) = -2.41, p = .018$). However, due to the fact that there were a total of six related t-tests performed, a Bonferroni correction should be applied in order to account for the chance factor. Thus, the α-level becomes $\alpha = 0.05/6 = 0.00833$ and we cannot say that the difference in question 4 is statistically significant.

In order to analyze how students with different previous knowledge about binary heaps learned the subject, we divided students into two groups: Students who scored points in pre-test (26 students in both conditions) and students who did not score any points in pre-test (26 and 27 students in viewing and changing conditions, respectively). The results are shown in Table 2. None of the differences were statistically significant. For the NPK groups, the largest differences were in question 5 and in the overall difference $t(50) = -1,72, p < .1$ and $t(50) = -1.71, p < .1$, respectively. For the SPK, the largest one was question 4 $t(50) = -2.363, p < .05$, when $\alpha = 0.05/6 = 0.0083$.

5. DISCUSSION

Based on this study, we cannot confirm our hypothesis that ET level would have a significant effect on the learning outcomes. However, in every question, students in *changing* condition performed better or at least as well as students in the *viewing* condition (see the Table 1).

We divided students further into two groups in order to see if there are differences in students learning outcomes depending on their previous knowledge of binary heaps. This division indicated that the differences between conditions were larger when students had no previous knowledge of binary heaps. Although differences were not statistically significant, this seem to indicated that higher ET level of the visualizations may have greater impact on students learning results when they do not have previous knowledge.

There are at least three explanations why the study could not find significant differences: there are no detectable differences, the differences could not be detected due to uncontrolled variables or because of design issues in the treatments or in the pre-test and post-test (see below), or the sample size was not large enough in order to detect the differences. This needs to be clarified in the future research.

In order to analyze the reasons behind the insignificant results, we calculated the effect sizes of the treatment. The Cohen's effect size measure [2] for the whole sample, NPK sample and SPK sample respectively were, 0.28, 0.53 and 0.24, for the total difference between pre-test and post-test. Based on Cohen's classification [2], the effect size of the students without previous knowledge is *medium*. This indicates that the effect could be found to be significant with a larger sample size and it should have some practical significance. The effect of the treatment was *small* for the students who had previous knowledge and this affected on the over all effect size which was also *small*.

In addition, we analyzed the individual scores from each student on each question to determine any anomaly in the scores that might explain our results. Scores from question 6 indicate that the question might have been too easy for the students. Question 6 asked to draw binary tree representation from array presentation of the minimum binary heap. The reason why this question might have been too easy for students is that they had been just shown heap formation in either tree or array format in the visualizations and they seemed to have grasped that well. Moreover, questions which were formulated in a way that gave students a 50 percent chance to answer correctly (yes/no-questions) by guessing might have distorted the scores. In overall, students' performance in the post-test was very good. In the future, the test need to be made more difficult in order to tease out the differences.

When grading the question 4, there was a misconception of a heap's characteristic found. Several students assumed that the maximum heap's smallest value was found in the lowest and rightmost node in the heap's binary tree presentation. We need to investigate this misconception further in order to understand where this misconception has appeared and if the materials should be changed in order to prevent this misconception.

Table 2: The mean differences between pre-test and post-test for each question and for the total points between the condition for students with no previous knowledge (NPK) and students with some previous knowledge (SPK). The final row shows the averaged normalized difference in in total scores between pre- and post-test calculated for each student with formula $(p_{posttest} - p_{pretest})/(24 - p_{pretest})$. Standard deviations are in the parenthesis.

Question	Viewing NPK	Changing NPK	Viewing SPK	Changing SPK
1	2.62 (1.94)	3.08 (1.72)	0.62 (2.45)	1.00 (2.81)
2	2.73 (1.87)	3.31 (1.49)	1.46 (2.08)	1.04 (2.30)
3	3.73 (0.96)	4.00 (0.00)	0.77 (1.97)	1.30 (1.88)
4	2.15 (1.43)	2.50 (0.81)	1.38 (1.24)	2.22 (1.34)
5	2.31 (1.64)	3.00 (1.23)	2.62 (1.53)	2.48 (1.70)
6	3.73 (0.96)	3.73 (0.96)	2.38 (1.94)	2.41 (1.97)
Total	17.27 (5.61)	19.62 (4.20)	9.23 (5.67)	10.44 (6.29)
Normalized Total	0.72 (0.23)	0.82 (0.17)	0.60 (0.33)	0.64 (0.33)

6. CONCLUSION AND FUTURE WORK

We presented an empirical study which analyzed the hypothesis that ET framework can be used to predict performance differences when visualization are used in collaboration. We could not confirm this hypothesis, although results point to that direction especially for students without any previous knowledge.

As a future work, we will analyze the differences in the more difficult questions that were only present in the post-test. Furthermore, we will analyze students' behavior during the learning session from the recorded audio and screen capture. In this way, we can see if the level of engagement has an effect on the students learning process.

As pointed out in the discussion, there were some flaws in the design of the pre-test and post-test questions, which will be corrected and the experiment will be replicated during the spring 2007.

7. REFERENCES

[1] R. Ben-Bassat Levy, M. Ben-Ari, and P. A. Uronen. The Jeliot 2000 program animation system. *Computers & Education*, 40(1):15–21, 2003.

[2] J. Cohen. *Statistical power analysis for the behavioral sciences*. Academic Press, New York, 1977.

[3] C. Evans and N. J. Gibbons. The Interactivity Effect in Multimedia Learning. Accepted to Computers & Education, 2006.

[4] S. Grissom, M. McNally, and T. L. Naps. Algorithm visualization in CS education: comparing levels of student engagement. In *Proceedings of the First ACM Symposium on Software Visualization*, pages 87–94, June 2003.

[5] T. Hübscher-Younger and N. H. Narayanan. Constructive and collaborative learning of algorithms. *SIGCSE Bulletin*, 35(1):6–10, 2003.

[6] C. D. Hundhausen. Integrating Algorithm Visualization Technology into an Undergraduate Algorithms Course: Ethnographic Studies of a Social Constructivist Approach. *Computers & Education*, 39(3):237–260, 2002.

[7] C. D. Hundhausen and J. L. Brown. Designing, Visualizing, and Discussing Algorithms within a CS 1 Studio Experience: An Empirical Study. *Computers & Education*, In press.

[8] C. D. Hundhausen, S. A. Douglas, and J. T. Stasko. A Meta-Study of Algorithm Visualization Effectiveness. *Journal of Visual Languages and Computing*, 13(3):259–290, 2002.

[9] A. Korhonen, L. Malmi, P. Silvasti, V. Karavirta, J. Lnnberg, J. Nikander, K. Stålnacke, and P. Ihantola. Matrix — a framework for interactive software visualization. Research Report TKO-B 154/04, Laboratory of Information Processing Science, Department of Computer Science and Engineering, Helsinki University of Technology, 2004.

[10] N. Nagappan, L. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, and S. Balik. Improving the CS1 experience with pair programming. In *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, pages 359–362. ACM Press, 2003.

[11] T. L. Naps and S. Grissom. The effective use of quicksort visualizations in the classroom. *Journal of Computing Sciences in Colleges*, 18(1):88–96, 2002.

[12] T. L. Naps, G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, and J. Á. Velázquez-Iturbide. Exploring the Role of Visualization and Engagement in Computer Science Education. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, pages 131–152, New York, NY, USA, 2002. ACM Press.

[13] J. Roschelle. Designing for cognitive communication: Epistemic fidelity or mediating collaborating inquiry. In D. L. Day and D. K. Kovacs, editors, *Computers, Communication & Mental Models*, pages 13–25. Taylor & Francis, London, 1996.

[14] M. Scaife and Y. Rogers. External cognition: how do graphical representations work? *International Journal of Human-Computer Studies*, 45(2):185–213, 1996.

[15] D. D. Suthers and C. D. Hundhausen. An experimental study of the effects of representational guidance on collaborative learning processes. *Journal of the Learning Sciences*, 12(2):183–219, 2003.

[16] L. Williams, R. R. Kessler, W. Cunningham, and R. Jeffries. Strengthening the case for pair programming. *IEEE Software*, 17(4):19–25, 2000.

P3.

3

Laakso, M.-J., Myller, N., and Korhonen, A. (2009). Comparing learning performance of students using algorithm visualizations collaboratively on different engagement levels. Accepted for publication in *Journal of Educational Technology & Society*.

Comparing Learning Performance of Students Using Algorithm Visualizations Collaboratively on Different Engagement Levels

Mikko-Jussi Laakso
Department of Information Technology
University of Turku
22014 Turun Yliopisto
Turku, Finland
milaak@utu.fi
tel. +358 2 333 8672
fax. +358 2 333 8600

Niko Myller
Department of Computer Science and Statistics
University of Joensuu
P.O. Box 111
FI-80101 Joensuu
Joensuu, Finland
nmyller@cs.joensuu.fi
tel. +358 13 251 7929
fax. +358 13 251 7955

Ari Korhonen
Department of Computer Science and Engineering
Helsinki University of Technology
P.O. Box 5400
FI-02015 TKK
Espoo, Finland
archie@cs.hut.fi
tel. +358 9 451 3387
fax. +358 9 451 3293

Abstract

In this paper, two emerging learning and teaching methods have been studied: collaboration in concert with algorithm visualization. When visualizations have been employed in collaborative learning, collaboration introduces new challenges for the visualization tools. In addition, new theories are needed to guide the development and research of the visualization tools for collaborative learning. We present an empirical study, in which learning materials containing visualizations on different Extended Engagement Taxonomy levels were compared, when students were collaboratively learning concepts related to binary heap. In addition, the students' activities during the controlled experimental study were also recorded utilizing a screen capturing software. Pre- and post-tests were used as the test instruments in the experiment. No statistically significant differences were found in the post-test between the randomized groups. However, screen capturing and voice recording revealed that despite the randomization and instructions given to the students, not all of the students performed on the engagement level, to which they were assigned. By regrouping the students based on the monitored behavior, statistically significant differences were found in the total and pair average of the post-test scores. This confirms some of the hypothesis presented in the (Extended) Engagement Taxonomy.

Keywords: Algorithm Visualization, Algorithm Simulation, Collaborative Learning, Engagement Taxonomy

1 Introduction

Since its introduction, it has been hoped that Algorithm Visualization (AV) would solve problems related to learning of data structures and algorithms. However, empirical evaluations have yielded mixed results when determining the usefulness of such visualizations as teaching and learning aids over traditional methods (see the meta-analysis of the research on AV by Hundhausen et al. (2002)). Thus, researchers have sought explanations for the mixed results as well as better grounds to justify the use of visualizations in teaching. Hundhausen et al. (2002) concluded that the activities performed by the students are more important than the content of the visualization. This has led to the analysis of different engagement levels Naps et al. (2002) by ITiCSE Working Group that proposed *Engagement Taxonomy* (ET) to describe the various types of activities that students perform with visualizations and their effect on learning and Myller et al. (2008) have developed it further into *Extended Engagement Taxonomy* (EET).

Collaboration has become accepted and popular in Computer Science education. A good example is the benefits of pair programming (Nagappan et al., 2003; Williams et al., 2000; McDowell et al., 2003). Whilst visualizations are employed in collaborative learning, collaboration introduces new challenges for the visualization tools. For example, the exchange of experiences and ideas, and coordination of the joint work are needed when students are not working individually anymore (Suthers and Hundhausen, 2003). Furthermore, visualizations can provide a shared external memory that can initiate negotiations of meanings and act as a reference point when ideas are explained or misunderstandings are resolved (Suthers and Hundhausen, 2003). This implies that also new theories are needed to guide the development and research of the visualization tools for collaborative learning.

In this paper, the applicability of EET in collaborative use of visualizations has been studied. We test the impact of EET levels on the performance when visualizations are used in collaboration. We present an empirical study, in which learning materials containing visualizations on different EET levels were compared when student pairs were collaboratively learning concepts related to binary heap. The pairs had a mutual task to read through a tutorial including visualizations and answer questions related to the topic. Although, statistically significant differences were not detected in a previous study, the results indicated that the engagement level of the visualizations has an effect on the performance when students are working in pairs (Myller et al., 2007). Thus, we replicated that study in a different institution, and improved the settings in such a way that the detection of the statistically significant differences would be possible. In this paper, we report the results from the replication study conducted at the Helsinki University of Technology in which two groups of students were randomized to the computer lab sessions. Each session was randomly assigned to an EET level, either *changing* or *controlled viewing* (in the rest of the paper this can be also shortened to viewing when we are discussing about the groups), with the limitation that parallel sessions belonged to different conditions.

During the analysis of the screen and voice recordings collected in the study, it was detected that despite the randomization and instructions given to the students, not all of the students performed their learning on the expected EET level. This meant that although the tool allowed students to learn on a higher EET level, some of the students choose not to do so, but worked on a lower engagement level. Fortunately, the screen capturing and voice recording done during the students' learning process provided us a tool for noticing this and taking it into account in the analysis. Thus, in addition to the results from the study, we learned an important methodological lesson as well. Screen capturing and voice recording should be a standard procedure, because otherwise we cannot know for sure if the participants really do what we expect them to do.

In Chapter 2, we describe the relevant literature related to the engagement taxonomy and similar theories. In addition, we give an overview of the learning tool used in the experiments. Chapter 3 describes the research setting, i.e., the used pre- and post-tests, subjects, materials, and procedures. In Chapter 4, we report on the results. Finally, in Chapters 5 and 6, we make conclusions and highlight some future directions.

2 Previous Research

2.1 Visualizations and Engagement

As an attempt to describe the mixed results of previous research in AV usage (cf. (Hundhausen et al., 2002)) in learning and teaching of algorithms and data structures, Engagement Taxonomy (ET) was introduced by Naps et al. (2002). The central idea of the taxonomy is that the higher the engagement between the learner and the visualization, the higher the positive effects on learning outcomes. ET consists of six levels of engagement between the user and the visualization:

No viewing	There is no visualization to be viewed.
Viewing	The visualization is only looked at without any interaction.
Responding	Visualization is accompanied with questions, which are related to the content of the visualization.
Changing	Modification of the visualization is allowed, for example, by varying the input data set or algorithm simulation.
Constructing	Visualization of program or algorithm is created.
Presenting	Visualizations are presented to others for feedback and discussions.

ET has been used in the development of AV tools and several studies have utilized the framework and provided further support for it (see, e.g., Grissom et al. (2003); Naps and Grissom (2002)). However, the time to study the materials on different ET levels has commonly been an uncontrolled variable in the studies, meaning that students have had freedom to use as little or as much time as they wanted to. Thus, those students who have been studying with visualizations that are on the higher ET level have spent more time on the task. This, in turn, makes it questionable if the reason for better performance in the post-test is due to the additional time spent on studying or the higher ET level of the materials. In the experiment, which is presented in this paper, we controlled the time so that all the students needed to spend exactly the same amount of time on learning the topic.

There are also other studies which have shown that visualizations improve learning, without actually utilizing the ET framework in the design of the study (Ben-Bassat Levy et al., 2003). In addition to this, research in educational psychology and multimedia learning had also had similar results (Evans and Gibbons, 2006).

Myller et al. (2008) have proposed an extension to the ET called the *Extended Engagement Taxonomy* (EET). The idea of this extension is to let the designers and researchers of visualizations to use finer granularity of engagement levels in their tools and experimental designs. They provide the following engagement levels to be used together with the original ones: *controlled viewing*, *providing input*, *modification*, and *reviewing*. In this study, we will utilize the controlled viewing level in order to make a difference between the visualizations that can only be viewed by the student (EET level: *viewing*, e.g. static visualizations or animations with only a playing option) compared to those which can be controlled (EET level: *controlled viewing*, e.g. animations with VCR-like controls in order to step and play the animation both forwards and backwards).

2.2 Visualizations and Collaboration

From a more general perspective, there are studies that analyze the use of visualizations in collaboration. For instance, Suthers and Hundhausen (2003) have performed research in the area of scientific inquiry. They compared the effects of different representations (i.e., matrix, graph, and text) when students were collecting and analyzing data, hypotheses and their evidential relations. Their research showed that the form of the visualization and what kinds of interactions it drives have an effect on the collaboration process by making certain data and their relations more explicit or implicit.

Roschelle (1996) studied pairs of students using the learning environment of Newtonian physics and analyzed their learning outcomes as well as the process that led to those outcomes. During the study, it was recognized that learning tools and especially visualizations used in collaboration should focus more on supporting communication rather than presenting the underlying model as accurately as possible. Furthermore, Roschelle (1996) tells as the last lesson in his paper that, “one should design activities, which actively engage students in doing and encountering meaningful experiential feedback as a consequence of their actions”. Scaife and Rogers (1996) also identified the analysis of the interactions between external presentation and its users as a key research area for the future. All these points of view seem to support the applicability of ET/EET in the context of collaborative learning.

Although several AV tools have been developed and empirical studies carried out, the collaborative use of AV tools is researched very little. Myller et al. (2008) have studied the applicability of EET to describe differences in the learning process when visualizations are used during collaborative learning. They pointed out that when students were using visualizations on lower EET levels the interaction/engagement between students also dropped, meaning that students communicated and collaborated more when they were using materials on higher EET levels.

The work of Hundhausen (2002) is related to the collaborative aspects of AV construction and presentation. This work led into the development of a visualization tool, ALVIS, which supports construction and presentation of AVs in small groups (Hundhausen and Brown, 2008). Their results also indicate that ET is applicable in the context of collaborative learning, although it is not directly tested. Furthermore, Hundhausen (2005) has proposed a communicative dimensions framework in order to analyze the aspects of visualizations that affect communication between end-users. Hübscher-Younger and Narayanan (2003) developed a web-based system that allows students to post their own algorithm representations (e.g., text, pictures, animation, or multimedia) and discuss them on the web. The research concluded that the students who actively participated in this activity achieved higher grades than the passive students who might have only viewed and commented on others' presentations.

2.3 Other Algorithm Visualization Studies on Heap Data Structures

Stasko et al. (1993) utilized algorithm animations focusing on a pairing heap that was implemented as a binary tree. The results were disappointing: the animation group outperformed the control group but the differences were not high even on absolute scale, and the differences were not statistically significant. Moreover, they noted that using animations did not grant obvious learning benefits and they believe that algorithm animations benefit advanced students more than "novice students".

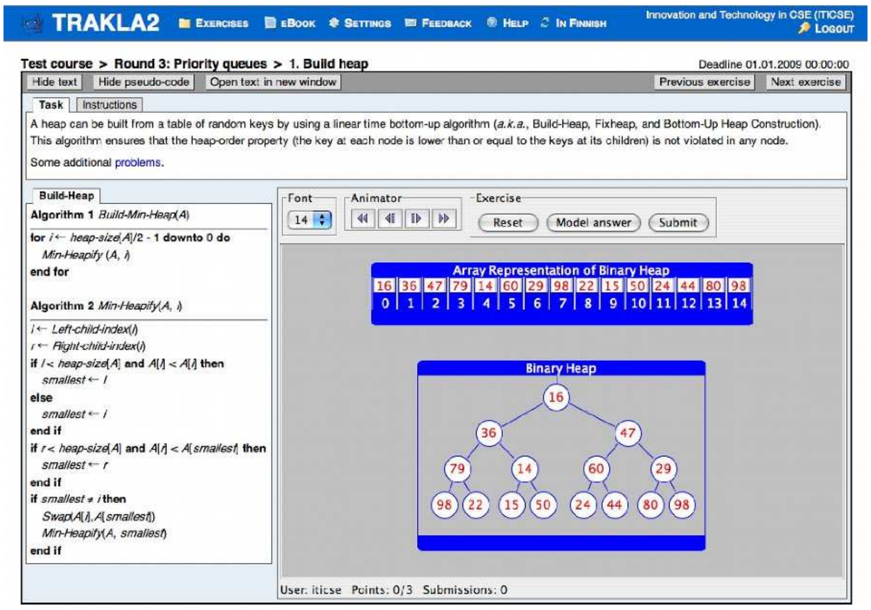
In 1996, Byrne et al. (1996) conducted algorithm animation research on binomial heap. The results were not statistical significant, either, and their findings supported the view that the benefits of animations are not that obvious, and careful task analysis is essential to determine in which situations animation can be helpful. Also Kehoe et al. (2001) studied the learning of binomial heap through animations in open lab sessions. They hypothesized that animations make complex algorithms more accessible and less intimidating and enhance students' motivation, interaction and learning. Their study, however, was inconclusive (they made hypotheses), and further empirical studies were suggested.

There are some differences between these studies and ours. Our students were novices with little or no previous knowledge on the topic, but they were not novices in using the visualization tool but had previous knowledge on how to use the tool and how to make sense of its visualization. However, students needed to study in our experiment concepts related to binary heap, which might be easier to understand and more accessible for novices compared to the pairing heap or the binomial heap. Furthermore, we used fixed time limits for the learning session meaning that all students needed to use exactly the same time to learn the topic, and we monitored their learning process in order to detect how they were learning.

2.4 TRAKLA2 Overview

TRAKLA2 is a practicing environment for *visual algorithm simulation exercises* (Korhonen et al., 2004) that can be assessed automatically. The system distributes individually tailored tracing exercises to students and provides feedback about students' solutions automatically. In visual algorithm simulation exercises, a student directly manipulates the visual representation of the underlying data structures (i.e., a student acts on the EET level *changing*). Thus, the student manipulates real data structures through GUI operations with the purpose of performing the same changes on the data structures the actual algorithm would do. An answer to an exercise is a sequence of discrete states of data structures, and the task is to perform the correct operations that will cause the transitions between each of the two consecutive states.

Each TRAKLA2 exercise page consists of a description of the exercise with links to other pages that introduce the theory and examples of the algorithm in question, instructions on how to interact with the GUI, code window, and an interactive Java applet. The current exercise set consists of over 40 assignments on basic data structures, sorting algorithms, search trees, hashing methods, and graph algorithms.



Test course > Round 3: Priority queues > 1. Build heap

Deadline 01.01.2009 00:00:00

A heap can be built from a table of random keys by using a linear time bottom-up algorithm (*a.k.a.*, Build-Heap, Fixheap, and Bottom-Up Heap Construction). This algorithm ensures that the heap-order property (the key at each node is lower than or equal to the keys at its children) is not violated in any node.

Some additional problems.

```
Algorithm 1 Build-Min-Heap(A)
for i ← heap-size[A]/2 - 1 downto 0 do
   Min-Heapify (A, i)
end for

Algorithm 2 Min-Heapify(A, i)
l ← Left-child-index(i)
r ← Right-child-index(i)
if l < heap-size[A] and A[l] < A[i] then
   smallest ← l
else
   smallest ← i
end if
if r < heap-size[A] and A[r] < A[smallest] then
   smallest ← r
end if
if smallest ≠ i then
   Swap(A[i],A[smallest])
   Min-Heapify(A, smallest)
end if
```

16	36	47	79	14	60	29	98	22	15	50	24	44	80	98
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

User: iticse Points: 0/3 Submissions: 0

Figure 1: TRAKLA2 exercise page. The student acts in EET level *changing* by solving the exercise in terms of swapping the data elements in the data structure(s).

Let us consider the exercise in Figure 1. The student is supposed to manipulate the visual representation(s) of the Binary Heap data structure by invoking context-sensitive *drag-and-drop operations*. The idea is to simulate the linear time BuildHeap algorithm. The manipulation can be done in either of the representations shown in the figure (i.e. the array or the binary tree representation). A key can be sifted up in terms of *swap operations* with its parent until the heap property is satisfied (the key at each node is smaller than or equal to the keys of its children). A single swap operation is performed by dragging and dropping a key in the heap on top of another key.

An exercise applet is initialized with *randomized input data*. The BuildHeap exercise, for example, is initialized with 15 numeric keys that correspond to the priority values. The student can *reset the exercise* by pressing the *Reset* button at any time. As a result, the exercise is reinitialized with new random keys. When attempting to solve the exercise, the student can *review the answer* step by step using the *Animator* panel. Moreover, the student can *Submit* the answer in which case the answer is assessed and immediate feedback is delivered. The feedback reports the number of correct steps out of the total number of steps in the exercise. This kind of automatic assessment is possible due to the fact that, again, the student is manipulating real data structures through the GUI. Thus, it is possible to *implement* the same algorithm the student is simulating, and execute it so that the algorithm manipulates the same data structures, but different instances, as the student just did. The assessment is based on comparison between these two different instances of data structures with each other.

An exercise can be submitted an unlimited number of times. However, a solution for a single instance of an exercise with certain input data can be submitted only once. In order to resubmit a solution to the exercise, the student has to reset the exercise and start over with new randomized input data. A student can also review a *Model answer* for each attempt. It is represented in a separate window as an algorithm animation accompanied with a pseudo code animation so that the execution of the algorithm is visualized step by step. The states of the model solution can be browsed back and forth using a similar animator panel as in the exercise. For obvious reasons — after opening the model solution — the student cannot submit a solution until the exercise has been reset and resolved with new random data.

TRAKLA2 visual algorith simulations and their instant feedback and model answer capabilities can also help students to collaborate with each other by providing shared external imagery and memory that can be processed together. Furthermore, they can increase the awareness of the students on each others abilities and knowledge (Collazos et al., 2007).

2.4.1 Previous Studies on TRAKLA2

In 2001, the first intervention study Korhonen et al. (2002) with three randomized groups A, B, and C (N_A=372,N_B=77,N_C=101) was performed. Students' behavior was monitored over the second year course in data structures and algorithms (DSA) lasting twelve weeks. The examination results of students using the TRAKLA learning environment (predecessor of TRAKLA2) were compared with those in the traditional classroom sessions. The results showed that, if the exercises are the same, there is no significant difference in the final examination results between students exercising on the web (group A) or in the classroom (group B). In addition, the commitment to the course (low drop-out rates), is almost equal in both versions of the course. However, if the exercises are more challenging (group C), there is a significant difference in the examination results, but the drop-out rate is significantly higher as well.

Laakso et al. (2005a) reported on another whole semester study, in which TRAKLA2 was introduced at the University of Turku. The students' learning results were compared between students, who used or did not use TRAKLA2, during a course on DSA. In addition, a survey-data (N = 100) was collected on the changes in students' attitudes towards web-based learning environments. The results showed that TRAKLA2 considerably increased the positive attitudes towards web-based learning. According to students' self-evaluations, the best learning results were achieved by combining traditional and web-based exercises. In addition, the overall student performance was clearly better than in 2003 when only in class pen-and-paper exercises were used.

In 2005, the 2001 and 2004 studies were repeated at the Helsinki University of Technology (HUT) and at the University of Turku (UTU) during the spring semester (Laakso et al., 2005b). The students (N = 133 + 134) were divided into two randomized exercise groups in both universities. The first group started their exercises on the web with the TRAKLA2 learning environment while the second group did their exercises in classroom sessions. In order to prevent the high drop-out rates (see, group C in 2001), however, the same learning experience were provided for all the students. At the midpoint of the course, the treatment for the students was changed. The first group continued in the class room and the second group on the web. Moreover, the same attitude survey, which carried out at UTU in 2004, was administered in both of the aforementioned universities.

The study concluded that it is good to introduce easy and guided exercises at the very beginning of the course. In addition to this, there is an emerging need for both web-based and classroom exercises. The recommended way to introduce the web-based exercises in DSA courses is by combining these two approaches. There is a set of exercises that are more suitable to be solved and automatically assessed on the web while the rest of the exercises are more suitable for traditional classroom sessions. More detailed information about this repetition study can be found in Laakso et al. (2005b).

The above studies were whole semester studies, in which the focus was on students' overall performance and drop-out rates. The difference between the treatments were in learning settings: the control groups were in classroom while the treatment groups were on the web. However, the learning objectives were the same for all groups, i.e., the exercises were algorithm simulation exercises. In addition, we studied the students' attitudes towards web based learning environments.

In contrast to the above studies, Myller et al. (2007) conducted an experimental study focusing on engagement taxonomy in fall 2006 at University of Turku. In the study, the learning outcomes of the students, who learned in collaboration by using visualization on different engagement levels were compared. There were 52 students in the treatment group (EET level: changing) and 53 students in the control group (EET level: controlled viewing), which sums up to 105 participants. The setup was a pre-test, treatment, post-test design. The post-test included the same questions as the pre-test, and additionally more difficult questions in order to see if the differences were apparent in them. The results indicated that the level of engagement had an effect on students' learning results in favor of the treatment group, although the differences were not statistically significant. Especially students without previous knowledge seemed to learn more from using visualizations on higher engagement level. In this paper, we report on a replication of this study with minor changes in order to repair the flaws in the design of the pre-test and post-test as reported by Myller et al. (2007).

3 Experimental Setup

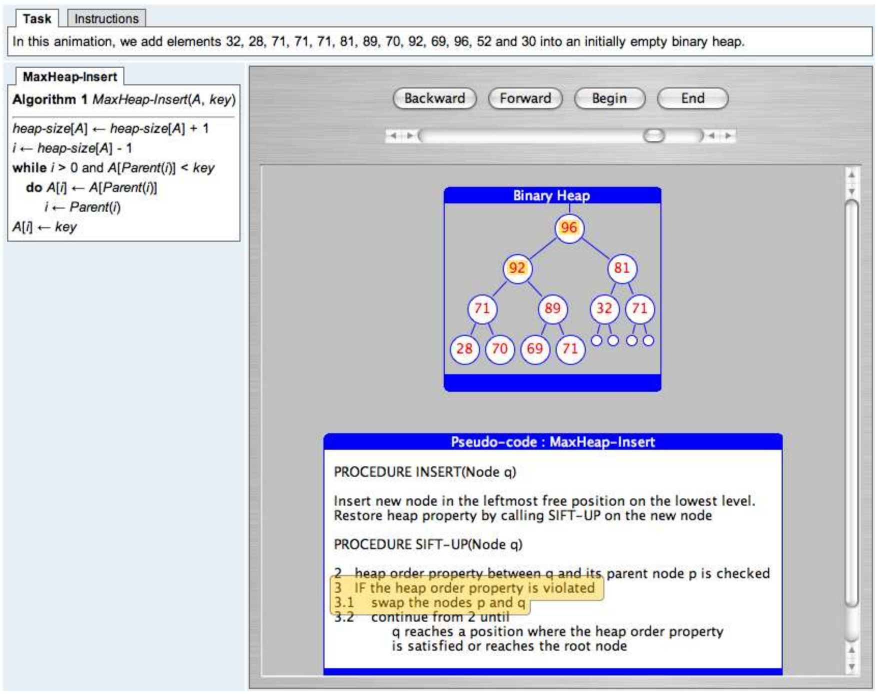
To summarize the previous sections, the collaborative use of AV tools has been studied only little, yet the need for this kind of research emerges from the increasing use of visualization tools in collaborative learning. We hypothesize that the EET framework can be used to predict performance differences when visualizations are used in collaboration. Previous research supports this view and our hypothesis is based on the previous research on TRAKLA2 and formulated as follows: Students using visualizations collaboratively on EET-level *changing* (i.e. in pairs) perform better compared to students using only visualization on EET-level *controlled viewing* (again in pairs).

In order to test our hypothesis, we carried out an experiment in which we compared the learning outcomes of students who were collaboratively using visualizations which were on different EET levels. Participants were (mostly first year) Computer Science major students on a data structures and algorithms course at the Helsinki University of Technology. We utilized TRAKLA2 (Korhonen et al., 2004) in order to provide students with algorithm simulation exercises that act on the EET level *changing* (treatment group). However, the students did not have the option to reset the exercise to obtain a new similar exercise with new input data, but they had to work with a fixed input data for each exercise during the whole session. The animations that the students used in *controlled viewing* condition (control group) were similar to those used in model answers provided by the TRAKLA2 system.

Quantitative results were analyzed with one-tailed t-test, ANOVA and χ^2-test depending on the nature of the data. We used the Bonferroni correction when applicable. The justification for using one-tailed t-test is based on the formulation of our hypothesis, which predicts that students using visualizations on EET-level *changing* perform better than students using visualization on EET-level *controlled viewing*. The hypothesis is based on the previous research in which it was found that student groups using visualizations on EET-level *changing* consistently performed better than student groups using visualization on EET-level *viewing* or *controlled viewing* although differences were not statistically significant Myller et al. (2007).

3.1 Method 1: Experimental Study

The study was a between-subject design with pre-test and post-test (dependent variable). We had one between-subject factor (independent variable): the highest available EET level of the visualizations in the learning materials, namely *controlled viewing* or *changing*. The unit of analysis was either a student or a pair of students depending on the measure. Each student answered the pre- and post-test individually, but all the observational data collected during the pair learning is not individual but the same for the pair. Moreover, we also report the average performance of the pair in the post-test and use it in the analysis.



In this animation, we add elements 32, 28, 71, 71, 71, 81, 89, 70, 92, 69, 96, 52 and 30 into an initially empty binary heap.

```
Algorithm 1 MaxHeap-Insert(A, key)
heap-size[A] ← heap-size[A] + 1
i ← heap-size[A] - 1
while i > 0 and A[Parent(i)] < key
  do A[i] ← A[Parent(i)]
     i ← Parent(i)
A[i] ← key
```

```
Pseudo-code : MaxHeap-Insert

PROCEDURE INSERT(Node q)

Insert new node in the leftmost free position on the lowest level.
Restore heap property by calling SIFT-UP on the new node

PROCEDURE SIFT-UP(Node q)

2   heap order property between q and its parent node p is checked
3   IF the heap order property is violated
3.1   swap the nodes p and q
3.2   continue from 2 until
          q reaches a position where the heap order property
          is satisfied or reaches the root node
```

Figure 2: Binary heap insert animation in the tutorial. The student acts on EET level *controlled viewing*. The user has VCR like buttons (Backward, Forward, Begin, End) to interact with the animation.

The learning materials contained textual materials that were the same for both conditions. In the *changing* condition, textual materials were accompanied with TRAKLA2 (Korhonen et al., 2004) algorithm simulation *exercises* related to the binary heap (see Figure 1). Student pairs in the *controlled viewing* condition were presented with *animations* about the operations of the binary heap that were similar to TRAKLA2 exercises (see Figure 2). In addition, student pairs in both conditions were given an exercise sheet that asked questions on binary heap that were supposed to be answered during the learning process. In this way, we tried to motivate the learning and make sure that the possible differences are due to controlled variable (level of engagement), and not because pairs in one condition performed cognitively more demanding activities or used more time on the tasks (Grissom et al., 2003; Hundhausen et al., 2002).

3.2 Method 2: Observational Study

The students' activities during the controlled experimental study were also recorded utilizing a screen capturing software. The recording accompanied by an audio track contained on-screen activity, i.e., mouse movements, keyboard typings, scrolling of the tutorial page back and forth in the browser window, as well as the conversation between the pair members.

The observed pairs were aware of being observed and we asked a permission to monitor them in advance. In this overt research method, we observed the students in their activities without intervention, i.e., by watching the recordings afterwards (Gall et al., 2006).

A detailed record of the events that occurred during the period of monitoring the students was produced. These events were categorized into the following four engagement levels according to the extended engagement taxonomy: *no viewing* (e.g., reading phase), *viewing* (e.g., watching figures), *controlled viewing* (e.g. watching of animations or model solution step-by-step with user controls) and *changing* (i.e., solving an algorithm simulation exercise). We separated passive *viewing* and more active *controlled viewing* from each other. In passive *viewing*, there was a still picture on the screen that we assumed the pair was watching. However, some of this time was spent to solve the given exercises on paper, as well. In *controlled viewing*, however, we knew that students were more actively involved with the animation as we required that they needed to control the animation by pressing VCR-like buttons to execute the animation backwards or forwards, and there were no pauses longer than 20 seconds between each action. The total time-on-task was measured from each four EET levels. Obviously, the students in *controlled viewing* condition (control group) did not spend time on *changing* mode. However, not all students in *changing* condition (treatment group) did either. Based on this analysis, we classified the students to groups based on their behavior.

3.3 Participants

Students were mainly first year students, however, some students from other years were also on the course. Students were randomized to the computer lab sessions and sessions were randomly assigned to each condition with the limitation that parallel sessions belonged to different conditions. The total number of participating students was 92. However, not all of them allowed to monitor their performance, nor were they willing to do pair work. In addition, in some of the workstations, the Java applet was not working properly. Moreover, we excluded foreign students from the study as they did not get the same treatment as the others due to the fact that their study materials were in a different language (i.e. English, while the original materials were in Finnish) and did not include animations nor algorithm simulation exercises, but they solved them by paper and pencil. Thus, the total number of analysis units (students) was 75 (*n*=75) divided into 7 small groups (3 control groups having *viewing* condition and 4 treatment groups having *changing* condition). The original number of lab sessions was 8, but the last one (that would have been control group) was the excluded English speaking group.

All students had been previously using TRAKLA2 during the course to complete three assignment rounds related to basic data structures (e.g. lists and stacks), algorithm analysis, sorting algorithms (i.e., insertion sort, quicksort, and mergesort), and binary tree traversing. Thus, all students should have been able to use TRAKLA2, understand its visualization, and know all its features that were needed to complete the assignments.

3.4 Materials

Pre-test consisted of the following questions. In the first question, the student were asked to define concepts *array*, *binary tree*, and *priority queue*. We assumed that the students are able to answer the first two as those concepts were already introduced in the course. The last concept and the rest of the questions were such that we assumed the participants do not have prior knowledge to answer them. However, we wanted to test whether they have some prior knowledge, e.g., due to taking the course already in the previous year (without passing it). The second question was, if a given array is a heap and the third, whether an ordered array is a heap or not. In addition, we asked the students to describe where the smallest value in a minimum binary heap (question 5) and maximum binary heap is located (question 6), respectively. Finally, we asked them to write down a given binary heap's heap property (question 7). The third question asked the students to draw the binary tree representation of the minimum binary heap, which was given in an array presentation, in the previous question.

The post-test consisted of the following questions. The pre-test and post-test included two questions which were exactly the same. The first question in the pre-test was omitted from the post-test. However, the questions 2, 3, 4, 5, 6 and 7 were the same in both (but the numbering started from 1 in the post-test). In addition, participants needed to do similar exercises that they did in the lab session. One of these was insertion of new items into an initially empty maximum binary heap (question 7 in the post-test). The question 8 asked participants to remove two smallest items from a minimum binary heap. Finally, we gave a pseudo-code example of a recursive `MAX-HEAPIFY` procedure and asked several questions, such as for which algorithm one can apply this procedure (question 9). This was a multiple choice question with four alternatives of which the last three were applicable: `Heap-Insert`, `Heap-Exctract-Max`, (linear-time) `BuildHeap`, and `HeapSort`. In addition, we asked them to describe and give an example execution (line-by-line) of what this procedure does and how (question 10). Question 11 requested the participants to provide an example which shows the recursive nature of the algorithm. The code example did not have a complete implementations for how to inquire the left and right child of a node in a complete binary tree implemented as an array. The task was to write this code (e.g., `LEFT(i) = 2i` and `RIGHT(i) = 2i+1`) (questions 12). Finally, they needed to analyze the worst case time complexity of `MAX-HEAPIFY` (question 13).

3.5 Procedure

Study was performed halfway through the course at the computer lab sessions that lasted for 2 hours. There were a total of 4 + 3 sessions, and they were run on two days in two following weeks. On each day, there were two times two sessions with different conditions running simultaneously. On the second day, there were also 4 sessions, but only 3 of them were included in this study as the last one was the excluded session given in English.

In the beginning of the session, students took the individual pre-test, in which they needed to answer questions related to binary heaps in 15 minutes. After this, they freely formed pairs with their peers and gave their consent to participate in the experiment and to be monitored during the experiment. If there was an odd number of students, one group consisted of 3 students. Each pair was allocated to a single computer.

After the pre-test, students had 45 minutes to go through the learning materials of their condition and complete paper-and-pencil exercises together. The collaboration was monitored by recording their talking and capturing their activities on the computer screens. After the 45 minutes the paper-and-pencil exercises were collected and the session ended with an individual post-test. The students were given 30 minutes to answer the questions in the post-test.

Each question in the pre- and post-tests was analyzed in a scale from 0 and 4. Zero points meant less than 25 percent of the answer was correct in the answer, and each point meant a 25 percent increase in the correctness of the answer.

4 Results

4.1 Randomized Treatment and Control Groups

In this section, we report the results as they were obtained by using the randomized treatment groups (42 students) and control groups (33 students) (*n*=75).

4.1.1 Previous Knowledge and Motivation

All the information related to the previous knowledge of the students could be determined only through post-hoc analysis, and thus, we could not make sure before-hand that the randomization did not introduce any bias to the experimental settings. Table 1 represents the students' previous knowledge in Computer Science and Mathematics for both groups. The first column shows the pre-test scores for the topics studied in the experiment. The column "Prog. Course Results" shows the students' average grades from a previous programming course. The average number of CS and Math credits units (each credit unit equals to about 30 hours of work) obtained are shown in the next columns, respectively. The difference between groups in the previous programming course grades is approaching statistical significance (*t*(73)=-1.94, *p*=0.056). Other differences are statistically insignificant.

	Pre-test	**Prog. Course Grade**	**CS**	**Math**
Control (33)	9.27 (6.87)	2.61 (1.77)	10.72 (16.77)	9.13 (9.33)
Treatment (42)	8.57 (5.04)	3.36 (1.57)	10.44 (14.80)	8.34 (6.87)

Table 1: Previous knowledge of the students on Heap data structure, and in CS and Math.

Table 2 shows the results from a motivational questionnaire filled in by the students. The questions were answered in a 7-degree Likert-scale and they were as follows:

- Q1. How useful do you regard this course for your working career?
- Q2. Do you expect that the on-line learning will help your learning of the course content?
- Q3. How well do online exercises fit into this course?
- Q4. How useful have the on-line learning tools and materials been in your previous courses?

	Q1	**Q2**	**Q3**	**Q4**
Control	4.84 (1.25)	4.78 (1.18)	5.38 (1.01)	4.94 (1.39)
Treatment	5.12 (1.33)	5.24 (1.14)	5.88 (1.05)	5.59 (1.30)

Table 2: Motivation of students based on a questionnaire (questions Q1 to Q4 are discussed in the text).

There were no statistically significant differences between the groups in any of the questions in the motivational questionnaire.

4.1.2 Post-test results

In the post-test, we used the same questions as in the pre-test and in addition to this seven more demanding questions. In the questions that were the same as in the pre-test, control and treatment group received on average 16.88 points (*st.dev.* 4.34) and 17.38 points (*st.dev.* 4.32), respectively. When comparing the pre- and post-test scores on the same questions within the group, statistically significant differences were found in both groups' total scores using pairwise t-test (Control: $t(33)=-13.48$, $p<.001$, Treatment: $t(42)=-25.71$, $p<.001$) (see the Table 1 for average pre-test scores and standard deviations). This means that both groups had learned the subject, which seems obvious when they spent 45 minutes to learn the topic.

When the points from all the questions were summed together the control group received on average a total of 30.79 points (*st.dev.* 6.99) and the treatment group 31.55 (*st.dev.* 6.29) points out of 52 points. There were no statistically significant differences found between the post-test scores.

We further calculated pair averages by taking the average of individual post-test scores of the pair. We treat this value as the learning outcome of a pair. The pair averages for control and treatment groups were 30.68 points (*st.dev.* 4.74) and 31.63 points (*st.dev.* 4.44), respectively. There were no statistically significant differences between the final scores or in any individual question scores.

4.2 Observational Study

In this section, we report the results as obtained by using a video analysis to match the students activities with the definition of treatment and control group. Based on the analysis, we regrouped students into different groups based on their behavior during the observation. We identified three groups based on their assignment to control and treatment groups and their behavior. Firstly, the students in the control group seemed to behave homogeneously and they watched the animations as expected. We will refer to this group with the name *Viewing C* (C as in Control). Secondly, we identified a group of students in the treatment condition, who behaved exactly the same as the control group by only watching the animations and not even once trying to do any algorithm simulation exercises. We will refer to this group with the name *Viewing T* (T as in Treatment). We will refer to all students who only viewed the animations (i.e. students in groups *Viewing C* and *Viewing T*) with the name *Viewing A* (A as in All. Thirdly, we found the students who behaved as we expected in the treatment group. These students solved algorithm simulation exercises at least one time but most often three to six times. We will refer to this group with the name *Changing T*. The division of the groups is illustrated in Figure 3.

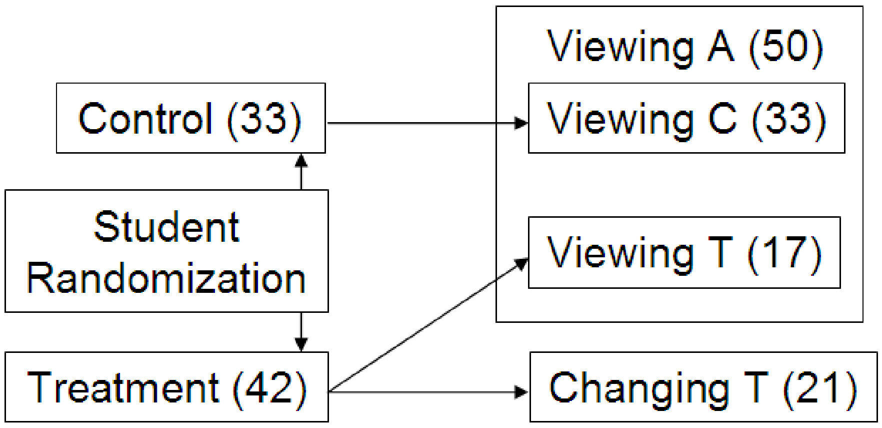


Figure 3: The division of the groups.

Based on the video analysis, we classified 33 students to the *Viewing C*, 17 student to the *Viewing T*, and 21 students to the *Changing T* (*n*=71). We needed to exclude four students from the analysis in this section due to technical problems when matching the students to correct videos. Two of the students would have belonged to the *Viewing T* and two to the *Changing T* groups.

In this section, we present two comparisons. Firstly, we analyze the data between three groups, namely *Viewing C*, *Viewing T* and *Changing T* because based on the original randomization and the video analysis these groups are distinct. However, when only the video analysis and groups' behavior is taken into consideration, we have only two groups, namely *Viewing A* and *Changing T*. Therefore, in order to provide a complete account of the results, we provide the analysis of both of these groupings. The validity, justifications and methodological implications of these groupings are further discussed in section **Error! Reference source not found.**.

4.2.1 Previous Knowledge and Motivation

The format of Table 3 is similar to the Table 1. None of the differences were statistically significant neither *Viewing C* vs. *Viewing T* vs. *Viewing T* nor *Viewing A* vs. *Changing T*. This was different compared to the original experimental design where there was a significant difference in favor of the treatment group in previous programming course grades.

	Pre-test	**Prog. Course Grade**	**CS**	**Math**
Viewing C	9.27 (6.87)	2.61 (1.77)	10.72 (16.77)	9.13 (9.33)
Viewing T	8.06 (4.49)	3.47 (1.46)	12.56 (21.04)	7.69 (6.63)
Viewing A	8.86 (6.14)	2.90 (1.71)	11.33 (18.10)	8.64 (8.46)
Changing T	9.29 (5.72)	3.14 (1.80)	10.43 (9.35)	9.67 (7.21)

Table 3: Previous knowledge of the students on Heap data structure, and in CS and Math.

Table 4 shows the results from the same motivational questionnaire that was also reported in the Table 2 for the experimental groups (See Section 4.1.1 for the description of the questions). None of the differences were statistically significant.

	Q1	**Q2**	**Q3**	**Q4**

Viewing C	4.84 (1.25)	4.78 (1.18)	5.38 (1.01)	4.94 (1.39)
Viewing T	5.00 (1.51)	5.25 (0.93)	5.81 (1.05)	5.44 (1.26)
Viewing A	4.90 (1.32)	4.94 (1.12)	5.52 (1.03)	5.10 (1.36)
Changing T	5.19 (1.33)	5.19 (1.36)	5.86 (1.11)	5.67 (1.43)

Table 4: Motivation of students based on a questionnaire (questions from Q1 to Q4 are discussed in Section 4.1.1).

4.2.2 Time Allocation between Engagement levels

Table 5 presents the distribution of the average times spent on each EET level. This was measured by watching the videos and marking times when the EET level changed from one to another, and then summing up the times on each EET level.

	No viewing	**Viewing**	**Controlled viewing**	**Changing**
Viewing C	47.45% (15.28)	38.26% (12.24)	14.29% (6.23)	0.00% (0.00)
Viewing T	49.45% (17.09)	37.82% (15.01)	12.73% (5.47)	0.00% (0.00)
Viewing A	48.13% (15.78)	38.11% (13.10)	13.76% (5.97)	0.00% (0.00)
Changing T	43.22% (19.20)	38.30% (15.84)	5.87% (6.03)	12.61% (1.98)

Table 5: The distribution of time (45 minutes) between EET levels.

Table 6 shows how many times students used materials on each EET level. For example, students in the control group used user-controlled visualizations (*controlled viewing*) 5 times on average, whereas students in the treatment group used them 2 or 3 times on average.

	No viewing	**Viewing**	**Controlled viewing**	**Changing**
Viewing C	6.76 (2.11)	7.82 (3.61)	5.15 (2.71)	0.00 (0.00)
Viewing T	7.18 (2.19)	7.53 (3.04)	5.29 (2.91)	0.00 (0.00)
Viewing A	6.90 (2.12)	7.72 (3.40)	5.20 (2.75)	0.00 (0.00)
Changing T	6.24 (1.73)	6.67 (3.20)	2.48 (2.56)	4.10 (1.61)

Table 6: The number of times each EET level was used.

4.2.3 Post-test results

The results of the post-test are presented in Table 7. When comparing the pre- and post-test scores within the group, statistically significant differences were found in both groups' total scores between pre- and post-tests when only same questions were compared with pairwise t-test (*Viewing C*: *t*(32)=-13.15, *p*<.001$, *Viewing T*: *t*(16)=-13.96, *p*<.001, *Viewing A*: *t*(49)=-18.09, *p*<.001, and *Changing T*: *t*(20)=-19.35, *p*<.001) (see the Table 3 for average pre-test scores and the *subtotal* in the Table 7 for the comparable average post-test scores and standard deviations).

	Viewing C	**Viewing T**	**Viewing A**	**Changing T**
Question 1	2.64 (1.58)	2.12 (1.65)	2.46 (1.61)	2.33 (1.80)
Question 2	1.76 (1.23)	1.82 (1.29)	1.78 (1.23)	2.19 (1.29)
Question 3	3.64 (1.08)	4.00 (0.00)	3.76 (0.89)	4.00 (0.00)
Question 4	2.39 (1.23)	2.18 (1.33)	2.32 (1.42)	2.33 (1.59)
Question 5	2.61 (1.43)	2.65 (1.58)	2.62 (1.47)	3.38 (0.92)
Question 6	3.85 (0.71)	3.76 (0.97)	3.82 (0.80)	4.00 (0.00)
Subtotal	16.88 (4.34)	16.53 (4.90)	16.76 (4.49)	18.24 (3.56)
Question 7	3.97 (0.17)	3.94 (0.24)	3.96 (0.20)	3.43 (1.29)
Question 8	3.33 (1.19)	3.65 (1.00)	3.44 (1.13)	3.76 (0.89)
Question 9	2.48 (0.87)	2.12 (0.78)	2.36 (0.85)	2.67 (0.91)
Question 10	2.09 (1.44)	2.41 (0.94)	2.20 (1.29)	2.62 (1.40)
Question 11	0.45 (1.25)	0.71 (1.45)	0.54 (1.31)	1.10 (1.70)
Question 12	1.30 (1.85)	0.18 (0.73)	0.92 (1.64)	1.24 (1.84)
Question 13	0.27 (0.45)	0.29 (0.99)	0.28 (0.67)	0.29 (0.46)
Total	30.79 (6.99)	29.82 (5.71)	30.46 (6.54)	33.33 (6.71)
Pair Average	30.68 (4.74)	29.88 (4.37)	30.42 (4.55)	33.45 (4.34)

Table 7: Post-test results (post-test questions were discussed in Section 3.4 and compostion of the groups in Figure 3).

Based on ANOVA, there were no statistically significant differences between *Viewing C*, *Viewing T* and *Changing T* groups in the post-test scores. When comparing the total values from the post-tests between *Viewing A* and *Changing T*, statistically significant differences were found in the total and pair average of the post-test scores by using one-tailed t-test ($t(69)=-1.73$, $p<0.05$) and ($t(31)=-1.97$, $p<0.05$), respectively.

5 Discussion

5.1 Interpretation of the Results

We presented an empirical study which analyzed whether the EET framework can be used to predict performance differences when algorithm visualizations are used in collaboration. Two randomized groups of students were involved in this study reading and answering questions related to a hypermedia tutorial presented on a web page. The control group used the algorithm visualizations on *controlled viewing* level, on which they had the opportunity to watch algorithm animations embedded in the tutorial. The treatment group interacted with the tutorial on *changing* level, on which they had the option to solve small algorithm simulation exercises and get feedback on their performance. In both groups, the students formed pairs and learned collaboratively about the binary heaps for 45 minutes during the 2-hour closed lab session. The analysis of the video material has showed that students were collaborating and discussing the subject matter during the learning process, therefore we are confident to say that students were truly learning collaboratively in both groups (Myller et al., 2008). The null hypothesis of the experiment was that there would be no significant statistical difference between the learning outcomes of the control and treatment group after the session.

Pre- and post-tests were used to analyze the performance. Each student answered these tests individually. There were no significant differences between groups if we analyzed only the pre-test scores. However, post-hoc analysis of some background variables revealed that there was almost a significant bias between the groups. The grades from the previous programming course were better in the treatment group than in the control group. Furthermore, based on the post-test results we could not reject the null hypothesis. This all was (at first) a counter-intuitive result, because a) it was against the theory that we were testing, b) it was against our previous findings and c) even the bias between the groups was in favor of the treatment group.

Fortunately, during the experimental study, we monitored the student pairs in a parallel observational study. After examining the video recordings, we realized that not all of the students in the treatment group were using the tutorial as expected. Some of the pairs did not solve the exercises, but only watched the model solutions instead. Thus, they were interacting with the tutorial only on *controlled viewing* level, not in *changing* level as expected. Based on this new evidence, we re-grouped the students. We regarded those students in the treatment group, not behaving on the changing level, belonging to a controlled viewing level. Interestingly, the aforementioned bias in previous programming course grades disappeared, and we found significant differences between the learning outcomes of the groups. Although there were no differences when only three groups were compared, the group working on changing level outperformed all student groups working on controlled viewing level in the total score of post-test. This was true both in the individual performance and the average performance of pairs. Thus, based on this study, we can reject the null hypothesis and confirm our previous findings that the level of engagement on which the students interact with the visualization tool has an influence on the learning. On changing level, they learned better than on controlled viewing level.

Stasko et al. (1993) hypothesize that “algorithm animations will not benefit novice students just learning a new topic as much as the animations will benefit more advanced students”, and moreover, that “the novice students would benefit more by actually constructing an algorithm animation rather than viewing a predefined one.” We can confirm these hypotheses. However, in this first hypothesis, we need to be careful in the definition of a “novice”. In our experiment, all students were exposed to TRAKLA2 before they attended the experiment. They solved similar exercises, but on different topics, a couple of weeks before the experiment took place. Thus, they were not “novices” when it comes to the “graphical notation” used in the experiment. Still, they were novices when it comes to the topic (i.e. they had not studied binary heaps earlier). Therefore, the conclusion is that the first hypothesis holds only if “novice” is defined to be a student who is not familiar with the used notation in the animations. One can still be a novice of the topic but understand the used notation, and benefit as much as more advanced students. Actually, it might even happen that the more advanced students cannot take the full advantage of this kind of learning material, and thus, perform worse, at least in relative scale (Myller et al., 2007). The confirmation of the second hypothesis is a direct outcome of our study in which the treatment group was “constructing an algorithm animation” in terms of changing the visualization, and they outperformed those students in the control group who just were “viewing a predefined” animation.

As discussed in the section on previous research, the learning time has not been a controlled variable in several previous studies, which have used the engagement level as the independent variable (Grissom et al., 2003; Naps et al., 2002; Hundhausen et al., 2002). Furthermore, it has been reported that students using visualizations on higher engagement levels have been motivated to spend more time on learning the topic. This has made it questionable if the time that students spend on learning the topic affects the learning results more than the engagement level, on which the visualization is used, and the engagement level affects only the amount of time students are willing to spend on learning the topic. In this study, we have shown that although we controlled the learning time and monitored students’ activities, the learning results are significantly different between engagement levels. This means that the engagement level has a direct effect on the learning results.

5.2 Methodological Considerations

Based on the results, screen capturing and voice recording should be a standard procedure because we cannot always know for sure if the participants really do what we expect them to do. Our study shows that we could not have obtained full understanding of the phenomenon without monitoring the students: not all of them performed on the expected engagement level even though we instructed them to do so. As we can see from our study, the conclusion would have been that we could not find any evidence that the EET level has an impact on learning, which would have been a false negative result. Thus, monitoring should be a standard procedure especially in large scale studies in which the researcher(s) cannot make sure by other means that the conditions remain constant within a group.

However, when using an observational design in the study, we need to pay attention to possible confounds that might affect our results. Due to the fact that in the observational study, we could not control the placement of participants into conditions, but they selected it themselves, this could have caused differences in the final results and there still might be background variables that we have not analyzed or detected affecting the results. However, as stated earlier, we did a post-hoc analysis of several background variables and detected that actually the re-grouping made the groups more similar on one aspect while keeping the other aspects unchanged. Thus, we are fairly confident that the observed differences are due to the claimed causes.

6 Conclusion and Future Work

Our results confirm that EET framework can predict performance differences also in collaborative use of visualizations. The results substantiate that there is a difference in learning results between viewing and changing modes. The findings of the observational study also explain why the original experimental design failed to reject the null hypothesis. This was due to the fact that students in the treatment group did not perform the learning tasks that we assumed them to do. Thus, they might have outperformed the control group in the experimental design if they only had performed in the changing mode.

From our point of view, the results emphasize the importance of engagement with visualizations, and we should promote systems that support different modes of engagement. The mere viewing of the algorithm animations is not enough, not even when there is a partner with whom to share the understandings and misunderstanding during the viewing of the visualization. Thus, we should, especially, design systems that act on the higher levels of the engagement taxonomy. For example, visual algorithm simulation exercises acting on the changing level produce better results compared to the viewing level. Furthermore, we should encourage the use of the systems on higher engagement levels in classrooms in order to achieve active and more student-centered learning. We hope this paper encourages teachers on different disciplines to try out visualization tools that enable higher engagement between the tool and the students especially in collaborative learning as this seems to increase the learning outcomes.

The future research challenge is to determine the importance and role of collaboration in the EET, i.e., can we repeat this experiment also in the case of individual learning? In this experiment, collaboration was used to encourage discussion in pairs and to collect better evidence of the real behavior in terms of screen capturing. The collaboration, however, has an influence on the performance as well. Thus, one research direction would be toward individual learning, but in a context that can still be monitored in order to prevent inconclusive results due to the fact that the individuals did not behave on the expected EET level.

7 Acknowledgements

This work was supported by the Academy of Finland under grant numbers 111350 and 210947. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Academy of Finland.

8 References

Ben-Bassat Levy, R., Ben-Ari & M., Uronen, P. A. (2003). The Jeliot 2000 program animation system. Computers & Education 40 (1), 15–21.

Byrne, M. D., Catrambone, R. & Stasko, J. T. (1996). Do algorithm animations aid learning? Technical Report GIT-GVU-96-18, Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA.

Collazos, C., Guerrero, L., Redondo, M., Bravo, C. (2007). Visualizing Shared-Knowledge Awareness in Collaborative Learning Processes. In J.M. Haake, S.F. Ochoa, and A. Cechich (Eds.), Proceedings of CRIWG 2007, Lecture Notes on Computer Science, vol. 4715, Springer Verlag Editions, pp. 56-71.

Evans, C. & Gibbons, N. J. (2007). The interactivity effect in multimedia learning. Computers & Education, 49 (4), 1147-1160.

Gall, M. D., Gall, J. P. & Borg, W. R. (2006). *Educational Research: An Introduction (8th Edition)*. Allyn & Bacon.

Grissom, S., McNally & M., Naps, T. L. (2003). Algorithm visualization in CS education: comparing levels of student engagement. In *Proceedings of the First ACM Symposium on Software Visualization*, ACM Press, 87–94.

Hübscher-Younger, T. & Narayanan, N. H. (2003). Constructive and collaborative learning of algorithms. SIGCSE Bulletin 35 (1), 6–10.

Hundhausen, C. D. (2002). Integrating Algorithm Visualization Technology into an Undergraduate Algorithms Course: Ethnographic Studies of a Social Constructivist Approach. Computers & Education 39 (3), 237–260.

Hundhausen, C. D. (2005). Using end-user visualization environments to mediate conversations: a ‘Communicative Dimensions’ framework. Journal of Visual Languages and Computing 16 (3), 153–185.

Hundhausen, C. D. & Brown, J. L. (2008). Designing, visualizing, and discussing algorithms within a CS 1 studio experience: An empirical study. Computers & Education 50 (1), 301–326.

Hundhausen, C. D., Douglas, S. A. & Stasko, J. T. (2002). A Meta-Study of Algorithm Visualization Effectiveness. Journal of Visual Languages and Computing 13 (3), 259–290.

Kehoe, C., Stasko, J. & Taylor, A. (2001). Rethinking the evaluation of algorithm animations as learning aids: An observational study. International Journal of Human-Computer Studies 54 (2), 265–284.

Korhonen, A., Malmi, L., Myllyselkä, P. & Scheinin, P. (2002). Does it make a difference if students exercise on the web or in the classroom? In *Proceedings of The 7th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'02*. ACM Press, 121–124.

Korhonen, A., Malmi, L., Silvasti, P., Karavirta, V., Lönnberg, J., Nikander, J., Stålnacke, K. & Ihantola, P. (2004). Matrix — a framework for interactive software visualization. Research Report TKO-B 154/04, Laboratory of Information Processing Science, Department of Computer Science and Engineering, Helsinki University of Technology.

Laakso, M.-J., Salakoski, T., Grandell, L., Qiu, X., Korhonen, A. & Malmi, L. (2005a). Multi-perspective study of novice learners adopting the visual algorithm simulation exercise system TRAKLA2. Informatics in Education 4 (1), 49–68.

Laakso, M.-J., Salakoski, T. & Korhonen, A. (2005b). The feasibility of automatic assessment and feedback. In Kinshuk, D. G. Sampson, P. Isafas, L. Rodrigues & P. Barbosa (Eds.) *Proceedings of Cognition and Exploratory Learning in Digital Age (CELDA 2005). IEEE Technical Committee on Learning Technology and Japanese Society of Information and Systems in Education*, IADIS Press, 113–122.

McDowell, C., Werner, L., Bullock, H. E. & Fernald, J. (2003). The impact of pair programming on student performance, perception and persistence. In *Proceedings of the 25th International Conference on Software Engineering*. IEEE Computer Society, 602–607.

Myller, N., Bednarik, R., Ben-Ari, M. & Sutinen, E. (2008). Extending the Engagement Taxonomy: Software

Visualization and Collaborative Learning. Submitted to the *Journal of Educational Resources in Computing (JERIC)*.

Myller, N., Laakso, M. & Korhonen, A. (2008). How Does Algorithm Visualization Affect Collaboration? Video Analysis of Engagement and Discussions. Submitted to the *Fourth International Computing Education Research Workshop (ICER 2008)*

Myller, N., Laakso, M. & Korhonen, A. (2007). Analyzing engagement taxonomy in collaborative algorithm visualization. In: Hughes, J., Peiris, D. R., Tymann, P. T. (Eds.), *ITiCSE '07: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*. ACM Press, 251–255.

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C. & Balik, S. (2003). Improving the CS1 experience with pair programming. In: *Proceedings of the 34th SIGCSE technical symposium on Computer science education*. ACM Press, 359–362.

Naps, T. L. & Grissom, S. (2002). The effective use of quicksort visualizations in the classroom. Journal of Computing Sciences in Colleges 18 (1), 88–96.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S. & Velázquez-Iturbide, J. Á. (2002). Exploring the Role of Visualization and Engagement in Computer Science Education. In: *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*. ACM Press, 131–152.

Roschelle, J. (1996). Designing for cognitive communication: Epistemic fidelity or mediating collaborating inquiry. In: *Day, D. L., Kovacs, D. K. (Eds.), Computers, Communication & Mental Models*. Taylor & Francis, London, 13–25.

Scaife, M., Rogers, Y. (1996). External cognition: how do graphical representations work? International Journal of Human-Computer Studies 45 (2), 185–213.

Stasko, J., Badre & A., Lewis, C. (1993). Do algorithm animations assist learning? an empirical study and analysis. In S. Ashlund, S., Henderson, A., Hollnagel, E., Mullet, K. & White, T. (Eds.) *Proceedings of the INTERCHI '93 Conference on Human Factors in Computing Systems*, 61–66.

Suthers, D. D. & Hundhausen, C. D. (2003). An experimental study of the effects of representational guidance on collaborative learning processes. Journal of the Learning Sciences 12 (2), 183–219.

Williams, L., Kessler, R. R., Cunningham, W. & Jeffries, R. (2000). Strengthening the case for pair programming. IEEE Software 17 (4), 19–25.

P4.

4

Korhonen, A., Laakso, M.-J., and Myller, N. (2009). How does algorithm visualization affect collaboration? Video Analysis of Engagement and Discussions. In the *Proceedings of the 5th International Conference on Web Information Systems and Technologies (WEBIST)*, pages 479–488.

HOW DOES ALGORITHM VISUALIZATION AFFECT COLLABORATION?
Video Analysis of Engagement and Discussions

Ari Korhonen
Department of Computer Science and Engineering, Helsinki University of Technology
P.O. Box 5400, FI-02015 TKK, Finland
archie@cs.hut.fi

Mikko-Jussi Laakso
Department of Information Technology, University of Turku, FI-22014 Turun Yliopisto, Finland
milaakso@utu.fi

Niko Myller
Department of Computer Science and Statistics, University of Joensuu, P.O. Box 111, FI-80101 Joensuu, Finland
niko.myller@cs.joensuu.fi

Keywords: Extended engagement taxonomy, Collaborative learning, Algorithm animation, Visual algorithm simulation.

Abstract: In this paper, we report a study on the use of Algorithm Visualizations (AV) in collaborative learning. Our previous results have confirmed the hypothesis that students' higher engagement has a positive effect on learning outcomes. Thus, we now analyze the students' collaborative learning process in order to find phenomena that explain the learning improvements. Based on the study of the recorded screens and audio during the learning, we show that the amount of collaboration and discussion increases during the learning sessions when the level of engagement increases. Furthermore, the groups that used visualizations on higher level of engagement, discussed the learned topic on different levels of abstraction whereas groups that used visualizations on lower levels of engagement tended to concentrate more on only one aspect of the topic. Therefore, we conclude that the level of engagement predicts, not only the learning performance, but also the amount of on-topic discussion in collaboration. Furthermore, we claim that the amount and quality of discussions explain the learning performance differences when students use visualizations in collaboration on different levels of engagement.

1 INTRODUCTION

Empirical evaluations have yielded mixed results when determining the usefulness of Algorithm Visualizations (AV) with empirical experiments. The meta-analysis by (Hundhausen et al., 2002) concluded that the activities performed by the students are more important than the content of the visualization. This has led to the proposition of *Engagement Taxonomy* by (Naps et al., 2002) to characterize the different levels of activities the students can perform with AV. The taxonomy is based on the Cognitive Constructivist learning theory (Hundhausen et al., 2002; Garrison, 1993; Piaget, 1977) and a student is supposed to achieve better learning results on higher engagement levels. Moreover, (Myller et al., 2008) have developed the taxonomy further by introducing *Extended Engagement Taxonomy* (EET), which describes the levels of engagement in finer level of detail. Furthermore, they have correlated the qualities of students' collaboration processes to different EET-levels, and therefore, extended the taxonomy into the direction of Social Constructivism (Palincsar, 1998; McMahon, 1997; Vygotsky, 1978).

Collaborative learning has become popular in Computer Science education (Beck and Chizhik, 2008; Teague and Roe, 2008; Valdivia and Nussbaum, 2007). Although visualizations have been employed in collaborative learning, collaboration introduces new challenges for the visualization tools. For example, the exchange of experiences and ideas, and coordination of the joint work are needed when students are no longer working individually (Suthers and Hundhausen, 2003). Furthermore, visualizations can provide a shared external memory that can initiate negotiations of meanings and act as a reference point when ideas are explained or misunderstandings are resolved (Suthers and Hundhausen, 2003). This implies that also new theories or extension of the previous ones are needed to guide the development and research of the visualization tools for collaborative learning.

In this paper, we study the use of AV in *collaborative learning*. We have utilized EET as a framework to test the impact of engagement levels on the learning process when the students work in pairs. In this experimental study, students collaborating on different engagement levels were compared with each other while they were learning concepts related to *binary heaps*. This is a follow-up study in a series of studies. The previous studies have shown that the engagement levels have a role to play in learning and showed that the use of visualizations on higher levels of engagement improves learning results (Laakso et al., 2009; Myller et al., 2007). However, this further investigation revealed new results that support the view that higher engagement levels have an effect not only on learning outcomes, but also on the amount of collaboration or discussion students have during the learning sessions. In other words, the engagement seems to have an effect not only on the student-content interaction, but also on the student-student interaction (see (Moore, 1989)). We hypothesize that these two together have influenced the students learning results.

Although a plethora of studies that concentrate on students' performance (Grissom et al., 2003; Naps and Grissom, 2002; Naps et al., 2002; Hundhausen et al., 2002) exist, we also need to understand the learning process and how the visualizations affect it (Hundhausen, 2002; Hundhausen and Brown, 2008). This information is essential when developing new systems in order to enhance students' learning with algorithm visualizations.

The structure of this paper is as follows: section 2 presents previous work on visualizations, engagement and interaction. The setup and design of the study are described in section 3. In section 4, the results are presented and they are further discussed in section 5. Finally, conclusions and future directions are given in section 6.

2 PREVIOUS WORK

2.1 Engagement

As an attempt to describe the mixed results of previous research in AV usage (Hundhausen et al., 2002) in learning and teaching of algorithms and data structures, Engagement Taxonomy (ET) was introduced by (Naps et al., 2002). The central idea of the taxonomy is that the higher the engagement between the learner and the visualization, the higher the positive effects on learning outcomes. ET consists of six levels of engagement between the user and the visualization:

No viewing – There is no visualization to be viewed.

Viewing – The visualization is only looked at without any interaction.

Responding – Visualization is accompanied with questions, which are related to the content of the visualization.

Changing – Modification of the visualization is allowed, for example, by varying the input data set or algorithm simulation.

Constructing – Visualization of program or algorithm is created.

Presenting – Visualizations are presented to others for feedback and discussion.

ET has been used in the development of AV tools and several studies have utilized the framework and provided further support for it (**?**, see, e.g.,)]Grissom2003, Grissom2002. There are also other studies which have shown that visualizations improve learning results, without actually utilizing the ET framework in the design of the study (Ben-Bassat Levy et al., 2003). In addition to this, research in educational psychology and multimedia learning have received similar results (Evans and Gibbons, 2007).

Although there is some anecdotal evidence on how the visualizations could affect collaborative learning process (Hundhausen, 2002; Hundhausen, 2005), there have been very few formal studies investigating it, especially from the point of view of engagement (Hundhausen and Brown, 2008). In this paper, we aim to research how the engagement between the learners and the visualization affects the interactions (i.e. collaboration and discussion) between learners.

(Myller et al., 2008) have proposed an extension to the ET called *Extended Engagement Taxonomy* (EET). The idea of this extension is to let the designers and researchers of visualizations to use finer granularity of engagement levels in their tools and experimental designs. They provide the following engagement levels to be used together with the original ones: *controlled viewing*, *providing input*, *modification*, and *reviewing*. In this study, we will utilize the controlled viewing level in order to make a difference between the visualizations that can only be viewed by the student (EET level: *viewing*, e.g. static visualizations or animations with only a playing option) compared to those which can be controlled (EET level: *controlled viewing*, e.g., animations with VCR-like controls in order to step and play the animation both forwards and backwards).

2.2 TRAKLA2

TRAKLA2 is a practicing environment for *visual algorithm simulation exercises* (Korhonen et al., 2003; Malmi et al., 2004) that are automatically assessed tracing exercises solved by a student in a web-based learning environment. The system distributes individually tailored exercises to students and provides instant feedback on students' solutions. In visual algorithm simulation exercises, a student directly manipulates the visual representations of the underlying data structures. Thus, the student actually manipulates real data structures through operations of the graphical user interface (GUI) with the purpose of performing the same changes on the data structures as the actual algorithm would perform. Each change leads the data structure to a new state. An answer to an exercise is a sequence of these states, and the task is to perform the correct operations that will simulate the running of the algorithm.

Each TRAKLA2 exercise consists of a description of the exercise accompanied with pseudo-code representation of the algorithm, and possibly support material that introduces the theory and examples of the algorithm in question, instructions on how to interact with the GUI, and an interactive Java applet that is utilized to enter the answer. The current exercise set consists of over 50 assignments on basic data structures, search structures, hashing methods as well as sorting and graph algorithms.

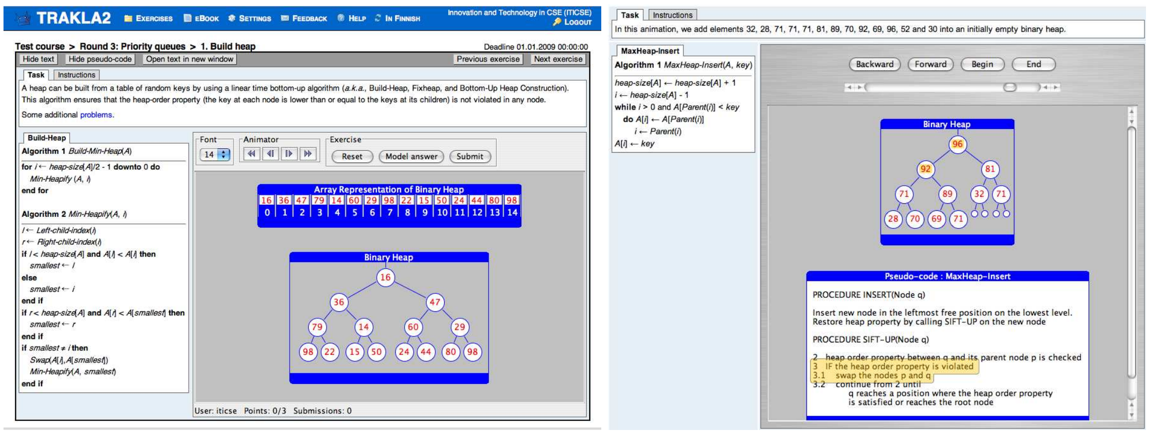
Example: Let us consider the exercise in Figure 1. The student is supposed to manipulate the visual representation(s) of the Binary Heap data structure by invoking context-sensitive *drag-and-drop operations*. The idea is to simulate the linear time BuildHeap algorithm. The manipulation can be done in either of the representations shown in the figure (i.e. the array or the binary tree representation). A key can be shifted up in terms of *swap operations* with its parent until the heap property is satisfied (the key at each node is smaller than or equal to the keys of its children). A single swap operation is performed by dragging and dropping a key in the heap on top of another key

An exercise applet is initialized with *randomized input data*. The BuildHeap exercise, for example, is initialized with 15 numeric keys that correspond to the priority values. The student can *reset the exercise* by pressing the *Reset* button at any time. As a result, the exercise is reinitialized with new random keys. When attempting to solve the exercise, the student can *review the answer* step by step using the *Animator* panel. Moreover, the student can *Submit* the answer for immediate assessment and feedback. The feedback reports the number of correct steps out of the total number of steps in the exercise. This kind of automatic assessment is possible due to the fact that the student is manipulating real data structures through the GUI. Thus, it is possible to *implement* the same algorithm the student is simulating, and execute it so that the algorithm manipulates the same data structures with same data, but different instances, as the student. Therefore, the assessment is based on the comparison of the two instances of the same data structures with each other.

An exercise can be submitted an unlimited number of times. However, a solution for a single instance of an exercise with certain input data can be submitted only once. In order to resubmit a solution to the exercise, the student has to reset the exercise and start over with new randomized input data. A student can also review a *Model answer* for each attempt. It is represented in a separate window as an algorithm animation accompanied with a pseudo code animation so that the execution of the algorithm is visualized step by step. The states of the model solution can be browsed back and forth using a similar animator panel as in the exercise. For obvious reasons — after opening the model solution — the student cannot submit a solution until the exercise has been reset and resolved with new random data.

2.3 Our Previous Studies on the Same Topic

The study reported in this paper belongs to a series of studies that have been run since autumn 2006 (Laakso et al., 2009; Myller et al., 2007). This is actually a follow-up video analysis of an experiment that we carried out in spring 2007 (Laakso et al., 2009). The objective of the experiment was to compare the learning outcomes of students who collaboratively used algorithm visualizations on two different EET levels, namely *controlled viewing* and *changing*. The results in sections 2.3.1 and 2.3.2 have already been reported and explained in more detail elsewhere (Laakso et al., 2009) but are given here in order to allow the discussion of them in relation to the findings that are reported in this paper in section 4. In the sections 2.3.1 and 2.3.2, the analysis was done for all the partic-



Test course > Round 3: Priority queues > 1. Build heap — Deadline 01.01.2009 00:00:00

A heap can be built from a table of random keys by using a linear time bottom-up algorithm (*a.k.a.*, Build-Heap, Fixheap, and Bottom-Up Heap Construction). This algorithm ensures that the heap-order property (the key at each node is lower than or equal to the keys at its children) is not violated in any node.

Some additional problems.

```
Algorithm 1 Build-Min-Heap(A)
for i ← heap-size[A]/2 - 1 downto 0 do
  Min-Heapify (A, i)
end for

Algorithm 2 Min-Heapify(A, i)
l ← Left-child-index(i)
r ← Right-child-index(i)
if l < heap-size[A] and A[l] < A[i] then
  smallest ← l
else
  smallest ← i
end if
if r < heap-size[A] and A[r] < A[smallest] then
  smallest ← r
end if
if smallest ≠ i then
  Swap(A[i],A[smallest])
  Min-Heapify(A, smallest)
end if
```

16	36	47	79	14	60	29	98	22	15	50	24	44	80	98
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

User: iticse Points: 0/3 Submissions: 0

In this animation, we add elements 32, 28, 71, 71, 71, 81, 89, 70, 92, 69, 96, 52 and 30 into an initially empty binary heap.

```
Algorithm 1 MaxHeap-Insert(A, key)
heap-size[A] ← heap-size[A] + 1
i ← heap-size[A] - 1
while i > 0 and A[Parent(i)] < key
  do A[i] ← A[Parent(i)]
     i ← Parent(i)
A[i] ← key
```

```
PROCEDURE INSERT(Node q)

Insert new node in the leftmost free position on the lowest level.
Restore heap property by calling SIFT-UP on the new node

PROCEDURE SIFT-UP(Node q)

2   heap order property between q and its parent node p is checked
3   IF the heap order property is violated
3.1   swap the nodes p and q
3.2   continue from 2 until
         q reaches a position where the heap order property
         is satisfied or reaches the root node
```

Figure 1: TRAKLA2 algoritm simulation exercise is on the left and corresponding model answer animation on the right.

ipants or groups of the same experiment that is reported in this paper. See section 3 for further description of the study design.

2.3.1 Learning Results

The pre- and post-test results for both conditions are given in Table 1. Based on the two-tailed t-test, the differences in the pre-test scores between conditions were not statistically significant meaning that the students' preliminary knowledge on the topic was similar. The differences in the post-test scores between conditions, both individual and group averages, were statistically significant based on the one-tailed t-test ($t(69) = -1.73, p < 0.05$ and $t(31) = -1.97, p < 0.05$, respectively). We used the one-tailed t-test to analyze the post-test scores because of our hypothesis that the treatment group was expected to perform better. This hypothesis was formed based on the Engagement Taxonomy (Naps et al., 2002) and the previous results in similar studies performed by others (Grissom et al., 2003; Hundhausen and Brown, 2008; Naps and Grissom, 2002) and us (Myller et al., 2007).

Table 1: The pre- and post-test results between conditions (standard deviations are given in parentheses) ($n = 71$).

	Pre-test	Post-test individual avg	Post-test group avg
Control	8.9 (6.1)	30.5 (6.5)	30.4 (4.6)
Treatment	9.3 (5.7)	33.3 (6.7)	33.5 (4.3)

2.3.2 Time Allocation between Engagement Levels

Table 2 presents the distribution of the average times spent on each EET level. This was measured by watching the videos and marking times when the EET level changed from one to another, and then summing up the times on each EET level. Based on this analysis, we made the final classification of groups into different conditions, because although some students were originally assigned to treatment condition, in which they were supposed to work on *changing* level, they never did, and therefore belonged to the control condition. This also shows that the amount of time that students spent on reading or looking at static images is almost the same in both groups and only the looking at the animations, which they could control, and the algorithm simulation exercises were used differently. In the control condition, the animations were the only active form of engagement whereas in treatment condition they also had the option of solving algorithm simulation exercises. The latter was more important due to the fact that this group used animations almost only for figuring out how they should simulate the algorithm.

Table 2: The distribution of time between EET levels (standard deviations are given in parentheses) ($n = 35$).

	Control	Treatment
No viewing	48.1% (15.8)	43.2% (19.2)
Viewing	38.3% (15.8)	38.1% (13.1)
Controlled viewing	13.8% (6.0)	5.1% (6.0)
Changing	0.0% (0.0)	12.6% (2.0)

Table 3: The number of times each EET level is entered (standard deviations are given in parentheses) ($n = 35$).

	Control	Treatment
No viewing	6.9 (2.1)	6.2 (1.7)
Viewing	7.7 (3.4)	6.7 (3.2)
Controlled viewing	5.2 (2.8)	2.4 (2.6)
Changing	0.0 (0.0)	4.1 (1.6)

Table 3 shows how many times students used materials on each EET level. For example, students in the control group used user-controlled animations (*controlled viewing*) 5 times on average, whereas students in the treatment group used them 2 or 3 times on average. This also shows the shift from the use of animations to the algorithm simulations in the treatment condition.

3 METHODOLOGY

This is a follow-up analysis for the quantitative study (Laakso et al., 2009), in which we showed that the use of higher engagement levels has an positive effect on the students' learning outcomes. Thus, the description of the experiment in this section is in many ways similar to the previous report. However, as we analyze the learning process — not its outcomes — the methodology is naturally different.

The objective in this study is to compare the learning processes of students who collaboratively used algorithm visualizations on two different EET levels, namely *controlled viewing* and *changing*. This is an observational study based on screen capture and audio recording analysis of students' interactions during the experiment. Students' activities were recorded utilizing a screen capturing software. The recordings were accompanied by an audio track and thus, contained on-screen activities, i.e., mouse movements, keyboard typings, scrolling of the tutorial page back and forth in the browser window, as well as the conversation between the pair members.

3.1 Participants

Students were mainly first year students, however, some students from other years were also on the course. All students had previously been using TRAKLA2 during the course to complete three assignment rounds related to basic data structures (e.g. lists, stacks and queues) and algorithm analysis, sorting and binary tree traversing. Thus, all students should have known how to use TRAKLA2, been familiar with its visualizations and all its features that were needed to complete the assignments.

Students were randomized to the computer lab sessions and sessions were randomly assigned to each condition with the limitation that parallel sessions belonged to different EET levels. The total number of participating students was 92. However, not all of them allowed to monitor their performance, nor were they willing to do group work. In addition, in some of the workstations, the Java applet was not working properly and there were problems in data capture. Thus, the total number of participants (students) was 71, divided into 7 groups (sessions). The original number of lab sessions was 8, but the last one (that would have been a control group) was excluded because it was an English speaking group, and the materials were mostly in Finnish.

The study was performed at the computer lab sessions that lasted for 2 hours, and they were run on two days in two consecutive weeks. Each day, there were two times two sessions with different conditions (control and treatment) running simultaneously. There were 10 to 15 participants in each session in both conditions. The external conditions, such as noise level, were similar in all sessions and based on the video and audio analysis it did not affect the learning process.

3.2 Procedure

In the beginning of the session, students took the individual pre-test, in which they answered questions related to binary heaps in 15 minutes. There were 9 simple questions about binary heaps, which could be answered with a few words, and one question asked students to draw a binary heap's tree representation. After this, they freely formed pairs with their peers and gave their consent to participate in the experiment and to be monitored during the experiment. If there was an odd number of students, one group consisted of 3 students. Each pair was allocated to a single computer.

After the pre-test, students had 45 minutes to go through the learning materials of their condition. The collaboration was monitored by recording their talking and capturing their activities on the computer screens. In addition, in this learning phase student were given three paper-and-pencil assignments. The session ended with an individual post-test. The students were given 30 minutes to answer the questions in the post-test. The post-test contained six questions which were the same as in the pre-test, and in addition to that, there were seven questions that were more difficult and comparable to the questions students needed to answer during the learning session.

Each question in the pre- and post-tests was analyzed on a scale from 0 and 4. Zero points meant less than 25 percent of the answer was correct in the answer, and each point meant a 25 percent increase in the correctness of the answer.

3.3 Method

In this overt research method, we observed the students in their activities, i.e., by watching the recordings afterwards (Gall et al., 2006). Participants were

Computer Science major students on a data structures and algorithms course at Helsinki University of Technology. The students worked in pairs, and they were aware of being observed. We asked a permission to monitor them in advance.

We utilized TRAKLA2 in order to provide the students with algorithm simulation exercises that act on the EET level *changing* (i.e, treatment group). However, the students did not have the option to reset the exercise in order to obtain a new similar exercise with new input data, but they had to work with a fixed input data for each exercise for the whole session. The animations that the students used on *controlled viewing* level (i.e., control group) were similar to the model answers provided by the TRAKLA2 system.

There was a total of 35 videos (about 45 minutes each), and we included three videos from both conditions into this analysis, in total six videos. From each video, we randomly selected a clip about 20 minutes long that contained activities on all applicable engagement levels. The videos were analyzed in five second time slots that were classified according to the following four factors.

The first factor classified the engagement level according to the extended engagement taxonomy: *no viewing* (e.g., reading phase), *viewing* (e.g., watching figures), *controlled viewing* (e.g., watching animations or model solutions step-by-step with user in control), and *changing* (i.e., solving an algorithm simulation exercise). However, if the students were solving the *paper-and-pencil exercises*, these episodes were classified into a separate class that was not used in the analysis. The second factor categorized each time slot based on audio analysis and determined whether the pair was having a conversation (or if they were silent). The third aspect specified the content of the conversation according to the following six categories: *algorithm and data structure (DS) behavior* (e.g., students discuss the features of binary heap), *the tool and its features* (e.g., students discuss how to use the tool), *exercise* (e.g., students discuss how to solve the exercise), *referring to the learning materials* (e.g., students are reading parts of the learning material out loud and then discussing that part of the materials), *on-topic* (i.e., students are discussing something that is related to the learning but does not belong to any other category) and *off-topic* (i.e., student are discussing something that does not relate to the learning process in any way).

Three different persons classified randomly selected videos with the restriction that each person analyzed at least one video from the control group and one from the treatment group.

4 RESULTS

In this section, we present the results of our study in which we analyzed the students' behavior during their learning process. Six groups were randomly selected (three groups from both conditions) from a total of 35 groups. We analyzed a 20-minute-long clip of screen capturing video and audio for each group in order to collect the amount of discussions, their contents, and the EET-level at each moment in order to understand the differences in the amount of discussions and their contents between the engagement levels.

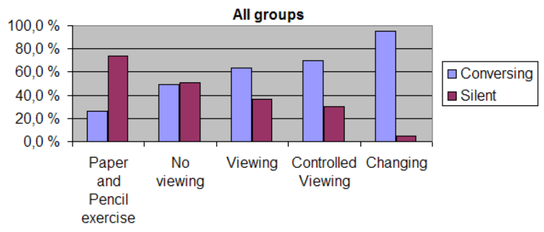


Figure 2: Distribution of activities in all groups on all EET-levels.

Figures 2, 3 and 4 show the distributions between the percentages of time that the students were conversing and silent. Based on the figures, one can see that the amount of conversation increases when the engagement level increases. This was also confirmed by using the χ^2-test on counts (all: $\chi^2(4) = 330.5, p < .001$, control: $\chi^2(4) = 84.1, p < .001$, treatment: $\chi^2(4) = 134.4, p < .001$), which showed that the engagement level has an effect on the amount of discussion, overall and in each condition.

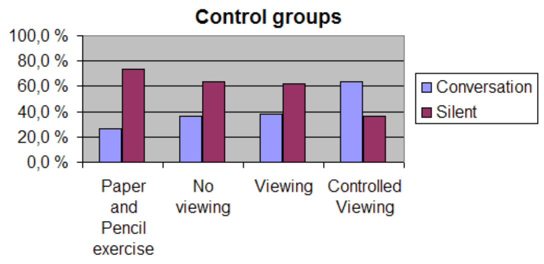


Figure 3: Distribution of activities for control groups on all EET-levels.

Pairwise comparison of the distributions on each EET level between conditions (see Figure 5) with the χ^2-test on counts showed that the distributions were different on levels *no viewing* ($\chi^2(1) = 9.2, p < .001$), *viewing* ($\chi^2(1) = 24.4, p < .001$) and *controlled viewing* ($\chi^2(1) = 21.4, p < .001$), but not when students were doing a paper and pencil exercise. The test could not be performed on the changing level as it was only available to treatment condition.

Table 4: Discussion content for all groups on all EET-levels.

	Alg. & DS behavior	**Code reading**	**Exercise**	**Referring to learning mat.**	**Tool**	**Coordination**	**On topic**	**Off topic**
Paper and pencil exercise	65.2%	0.9%	11.3%	0.0%	0.0%	15.7%	1.7%	5.2%
No viewing	39.2%	14.9%	5.4%	14.2%	4.1%	6.8%	10.1%	5.4%
Viewing	32.8%	24.8%	5.1%	4.4%	2.2%	7.3%	16.8%	6.6%
Controlled viewing	68.7%	19.4%	0.9%	3.3%	1.9%	3.8%	0.0%	1.9%
Changing	65.0%	0.0%	7.8%	0.0%	13.8%	1.8%	9.7%	1.8%

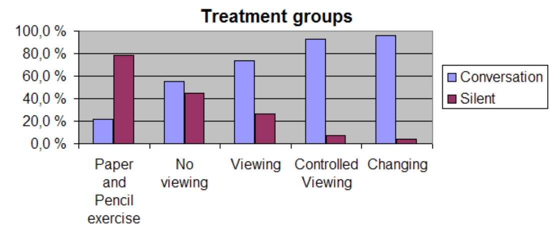


Figure 4: Distribution of activities for treatment groups on all EET-levels.

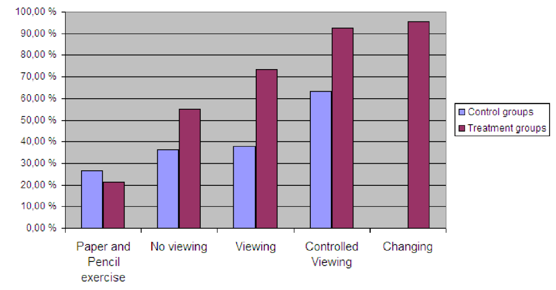


Figure 5: The percentages of time that students were discussing on each EET-level. The rest of the time students were silent. The control group does not have a value for the *changing* level, because it was unavailable for them.

Tables 4 and 5 show the distributions of the discussion contents on each engagement level and in each condition. When looking at the overall distribution, one can observe that the distributions of the discussion contents are similar on the *controlled viewing* and *changing* levels and when students are doing the paper and pencil exercises. Similarly, the distributions of *no viewing* and *viewing* seem more alike.

However, when the distributions between the conditions are compared, it can be seen that the *no viewing*, *viewing* and *controlled viewing* levels induce different kinds of discussions between the conditions. In control condition, the discussions are more related to the algorithm and data structure (DS) behavior, whereas in treatment condition larger proportions of the discussions on these levels are related to the pseudo code reading. In treatment condition, the *changing* level seems to be similar to the *controlled viewing* level and the paper and pencil exercise doing.

5 DISCUSSION

In this study, we have investigated the collaboration process when students were learning with visualization on different engagement levels. We can conclude that higher engagement with the visualization has a positive effect on students interaction with each other. Moreover, it seems that when students work on a larger number of engagement levels, their collaboration and communication is further improved.

Our results support the findings of (Hundhausen and Brown, 2008; Hundhausen, 2002), i.e., the higher engagement level between the visualization and learners increases the peer-to-peer (or student-student by (Moore, 1989)) communication. Students are more actively involved as the engagement level increases. Based on the results, we can say that if students work on higher engagement levels, their activities also positively change on lower levels. This phenomenon can be easily observed when we investigate the changes in the amount of discussion in the Figures 3 and 4 on *controlled viewing* and *viewing* between control and treatment groups. When students were working on changing level in the treatment group, the amount of silence dramatically decreased as the engagement level increased. At the two highest levels, the silence is practically absent. In control condition, the amount of silence decreases, but the change is smaller. For example, there is over 30% of the time when students are silent on *controlled viewing* level in control group while the time of being silent is well below 10% for the treatment group. The same difference is much more drastic in the *viewing-level* between the groups. Our understanding is that this is due to the fact that while students are solving a paper-and-pencil or TRAKLA2 algorithm simulation exercise, they realize that they cannot solve it. Therefore, they need to go back to the learning materials or the correspond-

Table 5: Discussion content for control and treatment groups on all EET-levels.

	Alg. & DS behavior	**Code reading**	**Exercise**	**Referring to learning mat.**	**Tool**	**Coordination**	**On topic**	**Off topic**
CONTROL								
Paper and pencil exercise	67.0%	0.9%	11.6%	0.0%	0.0%	13.4%	1.8%	5.4%
No viewing	25.7%	0.0%	11.4%	25.7%	14.3%	14.3%	2.9%	5.7%
Viewing	54.2%	0.0%	12.5%	4.2%	0.0%	8.3%	8.3%	12.5%
Controlled viewing	79.7%	7.4%	0.7%	4.7%	0.7%	4.1%	0.0%	2.7%
TREATMENT								
Paper and pencil exercise	0.0%	0.0%	0.0%	0.0%	0.0%	100.0%	0.0%	0.0%
No viewing	43.4%	19.5%	3.5%	5.3%	0.9%	4.4%	12.4%	5.3%
Viewing	28.3%	30.1%	3.5%	5.3%	2.7%	7.1%	18.6%	5.3%
Controlled viewing	42.9%	47.6%	1.6%	0.0%	4.8%	3.2%	0.0%	0.0%
Changing	65.0%	0.0%	7.8%	1.8%	13.8%	1.8%	9.7%	1.8%

ing animation in order to understand, how to solve the exercise. The reason that this is happening more in the treatment condition is the instant feedback that TRAKLA2 provides on each simulation, which helps students to understand when their mental models of the algorithms and data structures are not viable and they need to revise them. This example also indicates that the visiting of the engagement levels does not happen in any particular order, but can happen randomly.

(Teasley, 1997) has found that talking is correlated with better learning results. This, at least partially, explains why students learned better in the treatment condition compared to the control condition. The visualization helped them to discuss relevant topics in order to learn them. Because the topic that the participants were studying was unfamiliar to most of the participants, the conversations in the group aided students to better cope with the questions and problems that arose during the learning process. Therefore, we believe that pair-support is one of the key factors in enhancing students' learning, and it should be taken into account when designing and developing next generation learning tools and methods. Teasley has also found that *transactive reasoning* (Berkowitz and Gibbs, 1983) is strongly correlated with learning results, and in the future studies, we will also analyze the amount of transactive reasoning in the discussions.

In addition to the amount of discussion, we analyzed the discussion contents. Based on the analysis, we found that the students' discussions were related to the learned topic, otherwise there were no large differences. The only noticeable difference was the absence of *code reading* on the *changing* level. When we compared the distributions between conditions, there were more noticeable difference. In the treatment condition, the discussions related to the algorithm and data structure behavior and code reading were more balanced on levels *no viewing*, *viewing* and *controlled viewing* compared to the control condition, where students concentrated on the algorithm and data structure behavior and had very little discussion on the code. One could argue that the discussions on various levels of abstraction increased the students' understanding of the topic, and therefore, this could also be an explanation why students performed better in the post-test in the treatment condition.

6 CONCLUSIONS

Many studies related to the use of algorithm visualizations (AV) in teaching and learning have focused on the learning outcomes. On the one hand, (Extended) Engagement Taxonomy (EET) has been suggested to answer the question, if an AV system is effective in this respect or not. On the other hand, collaborative learning (CL) has proven to be an efficient teaching and learning method. However, very little is known about the interconnection between EET and CL.

We have investigated the use of AV in CL in many repeated studies. Our previous studies have confirmed that the engagement levels have a role to play in learning outcomes. The pairs of students that used AV on higher engagement levels performed better in the post-test than those pairs learning on lower engagement levels. The research in this paper has revealed that the amount of discussion in collaboration is also different between engagement levels, and increases as the engagement level increases.

Based on this study, EET not only predicts the increase in learning performance when student groups learn with visualization on higher engagement level, but also explains it by enabling students to have more discussions on topics that are relevant for learning. Thus, engagement goes hand in hand with collaboration so that the engagement taxonomy level has an influence over the collaborative learning process as well as the learning outcomes.

6.1 Future Directions

(Teasley, 1997) has found that *transactive reasoning* (Berkowitz and Gibbs, 1983) (TR) is strongly correlated with learning results. Transactive reasoning is discussion about one's own or collaboration partner's reasoning and logical thinking. TRAKLA2 exercises have interesting interconnections with the characterizations of TR categories. For example, Teasley describes *prediction* type TR as "explaining ..., stating a hypothesis about causal effects ... ." Moreover, the *feedback request* category can be characterized with a question: "Do you understand or agree with my position?"

Even though these do not correspond directly to TRAKLA2 exercises, the same elements are present in the exercise solving process. The student is supposed to *predict* each step in the algorithm simulation; and s/he receives instant *feedback* from the exercise. Thus, this kind of framework could function as a future testbed to explain good learning results that also individual learners get in the TRAKLA2 environment or in any other environment.

ACKNOWLEDGEMENTS

This work was supported by the Academy of Finland under grant numbers 111396. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Academy of Finland.

REFERENCES

Beck, L. L. and Chizhik, A. W. (2008). An experimental study of cooperative learning in CS1. In *SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education*, pages 205–209, New York, NY, USA. ACM.

Ben-Bassat Levy, R., Ben-Ari, M., and Uronen, P. A. (2003). The Jeliot 2000 program animation system. *Computers & Education*, 40(1):15–21.

Berkowitz, M. W. and Gibbs, J. C. (1983). Measuring the development of features in moral discussion. *Merill-Palmer Quarterly*, 29:399–410.

Evans, C. and Gibbons, N. J. (2007). The interactivity effect in multimedia learning. *Computers & Education*, 49(4):1147–1160.

Gall, M. D., Gall, J. P., and Borg, W. R. (2006). *Educational Research: An Introduction (8th Edition)*. Allyn & Bacon.

Garrison, D. R. (1993). A cognitive constructivist view of distance education: An analysis of teaching-learning assumptions. *Distance Education*, 14:199–211.

Grissom, S., McNally, M., and Naps, T. L. (2003). Algorithm visualization in CS education: Comparing levels of student engagement. In *Proceedings of the First ACM Symposium on Software Visualization*, pages 87–94. ACM Press.

Hundhausen, C. D. (2002). Integrating algorithm visualization technology into an undergraduate algorithms course: Ethnographic studies of a social constructivist approach. *Computers & Education*, 39(3):237–260.

Hundhausen, C. D. (2005). Using end-user visualization environments to mediate conversations: A 'Communicative Dimensions' framework. *Journal of Visual Languages and Computing*, 16(3):153–185.

Hundhausen, C. D. and Brown, J. L. (2008). Designing, visualizing, and discussing algorithms within a CS 1 studio experience: An empirical study. *Computers & Education*, 50(1):301–326.

Hundhausen, C. D., Douglas, S. A., and Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing*, 13(3):259–290.

Korhonen, A., Malmi, L., and Silvasti, P. (2003). TRAKLA2: a framework for automatically assessed visual algorithm simulation exercises. In *Proceedings of Kolin Kolistelut / Koli Calling – Third Annual Baltic Conference on Computer Science Education*, pages 48–56, Joensuu, Finland.

Laakso, M.-J., Myller, N., and Korhonen, A. (2009). Comparing learning performance of students using algorithm visualizations collaboratively on different engagement levels. Accepted to the Journal of Educational Technology & Society.

Malmi, L., Karavirta, V., Korhonen, A., Nikander, J., Seppälä, O., and Silvasti, P. (2004). Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. *Informatics in Education*, 3(2):267 – 288.

McMahon, M. (1997). Social constructivism and the world wide web – a paradigm for learning. In *Proceedings of the ASCILITE conference*, Perth, Australia.

Moore, M. G. (1989). Editorial: Three types of interaction. *The American Journal of Distance Education*, page 16.

Myller, N., Bednarik, R., Ben-Ari, M., and Sutinen, E. (2008). Applying the Extended Engagement Taxonomy to Collaborative Software Visualization. Accepted to the ACM Transactions on Computing Education.

Myller, N., Laakso, M., and Korhonen, A. (2007). Analyzing engagement taxonomy in collaborative algorithm visualization. In Hughes, J., Peiris, D. R., and Tymann, P. T., editors, *ITiCSE '07: Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, pages 251–255, New York, NY, USA. ACM Press.

Naps, T. L. and Grissom, S. (2002). The effective use of quicksort visualizations in the classroom. *Journal of Computing Sciences in Colleges*, 18(1):88–96.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., and Velázquez-Iturbide, J. Á. (2002). Exploring the role of visualization and engagement in computer science education. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, pages 131–152, New York, NY, USA. ACM Press.

Palincsar, A. S. (1998). Social constructivist perspectives on teaching and learning. *Annual Review of Psychology*, 49:345–375.

Piaget, J. (1977). *The Development of Thought: Equilibration of Cognitive Structures*. Viking, New York.

Suthers, D. D. and Hundhausen, C. D. (2003). An experimental study of the effects of representational guidance on collaborative learning processes. *Journal of the Learning Sciences*, 12(2):183–219.

Teague, D. and Roe, P. (2008). Collaborative learning: towards a solution for novice programmers. In *ACE '08: Proceedings of the tenth conference on Australasian computing education*, pages 147–153, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

Teasley, S. (1997). Talking about reasoning: How important is the peer in peer collaboration. In Resnick, L., Säljö, R., Pontecorvo, C., and Burge, B., editors, *Discourse, Tools and Reasoning: Essays on Situated Cognition*, pages 361–384. Springer, New York.

Valdivia, R. and Nussbaum, M. (2007). Face-to-face collaborative learning in computer science classes. *International Journal of Engineering Education*, 23:434–440(7).

Vygotsky, L. S. (1978). In Cole, M., John-Steiner, V., Scribner, S., and Souberman, E., editors, *Mind in Society*. Harvard University Press, Cambridge, Mass.

P5.

5

Myller, N. (2007). Automatic generation of prediction questions during program visualization. *Electronic Notes in Theoretical Computer Science*, 178:43–49. (Proceedings of the Fourth Program Visualization Workshop).

Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science 178 (2007) 43–49
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Automatic Generation of Prediction Questions during Program Visualization

Niko Myller[1] ,[2]

Department of Computer Science
University of Joensuu
Joensuu, Finland

Abstract

Based on previous research, it seems that the activities performed by and the engagement of the students matter more than the content of the visualization. One way to engage students to interact with a visualization is to present them with prediction questions. This has been shown to be beneficial for learning. Based on the engagement taxonomy and benefits of the question answering during the algorithm visualization, we propose to implement an automatic question generation into a program visualization tool, Jeliot 3. In this paper, we explain how the automatic question generation can be incorporated into the current design of Jeliot 3. In addition, we provide various example questions that could be automatically generated based on the data obtained during the visualization process.

Keywords: Program visualization, Engagement taxonomy, Automatic question generation.

1 Introduction

According to Hundhausen et al. [6], the activities performed by and the engagement of the students matter more than the content of the visualization. Thus, a research program has been laid down in which the level of engagement (*engagement taxonomy*) and its effects on learning with algorithm or program visualization are being studied [13]. One of the ways to engage students to interact with a visualization is to present them with questions, which ask the students to predict what happens next in the execution or visualization (level 3: responding) [12]. This has been shown to be beneficial for learning as well [3,11]. Furthermore, interaction and question answering during learning have been found to have a positive influence on problem-solving ability in the given domain [5]. In addition to the benefits

[1] I want to thank Erkki Sutinen and Moti Ben-Ari for their helpful comments and guidance during the research work, and Andrés Moreno and Roman Bednarik for fruitful discussions related to the topic of this paper. Part of this work was carried out during author's visit to Massey University, Palmerston North, New Zealand.

[2] Email: `firstname.lastname@cs.joensuu.fi`

found in the literature, in my work in progress, I am trying to show that actually the engagement taxonomy has a linkage to collaboration. The hypothesis in this research is that the higher the engagement level the larger the positive impact on collaboration.

Based on the found benefits of the question answering during visualization, we propose to implement an automatic question generation into a program visualization tool, *Jeliot 3* [9]. In this paper, we explain how the automatic question generation can be incorporated into the current design of Jeliot 3. In addition, we provide various example questions that could be automatically generated based on the data obtained during the visualization process. Finally, conclusions and future directions are presented.

2 Jeliot 3

Jeliot 3 is a program visualization system that visualizes the execution of Java programs [9]. It has been designed to support the teaching and leaning of introductory programming. Jeliot visualizes the data and control flow of the program. In a classroom study, it was found that especially the mid-performers benefited from the use of Jeliot while the performance of others was not harmed [1].

We describe the structure of Jeliot 3 in Figure 1 in order to explain in the next section how the automatic question generation fit into the current design. The user interacts with the user interface and creates the source code of the program (1). The source code is parsed and checks are performed before the actual interpretation by *DynamicJava*, a Java interpreter (2). During the interpretation, a representation of the program's execution, MCode, is extracted (3). MCode is assembly like language in which each line represents a single instruction that contains instruction type, instruction identifier (can be used as a reference), variable number of operands (references to other instruction ids, type and value) and the instruction's location in source code [8]. MCode is interpreted and the directions are given to the visualization engine (4 and 5). The user can control the animation by playing, pausing, rewinding or playing step-by-step the animation (6). Furthermore, the user can give input to the program executed by the interpreter (6, 7 and 8). For further information related to the internal structure of Jeliot the reader is pointed to [8,10].

3 Automatic Question Generation in Jeliot 3

The steps of question generation are: 1) information collection, 2) question formation, and 3) its presentation during the animation. In order to generate prediction questions, we need to collect information related to the program interpretation. We list here items that are needed to generate a question:

- Type of Expression, so that different question text is generated for assignment expression compared to method call.
- Instruction identifier to ensure that the question is popped up at right time.

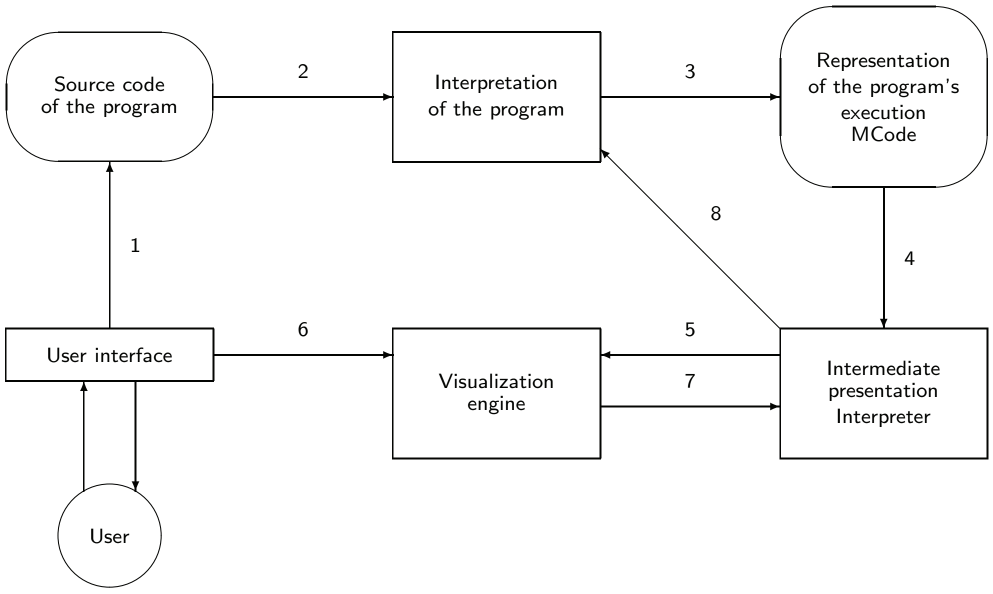

Source code of the program
Interpretation of the program
Representation of the program's execution MCode
User interface
Visualization engine
Intermediate presentation Interpreter
User
1
2
3
4
5
6
7
8

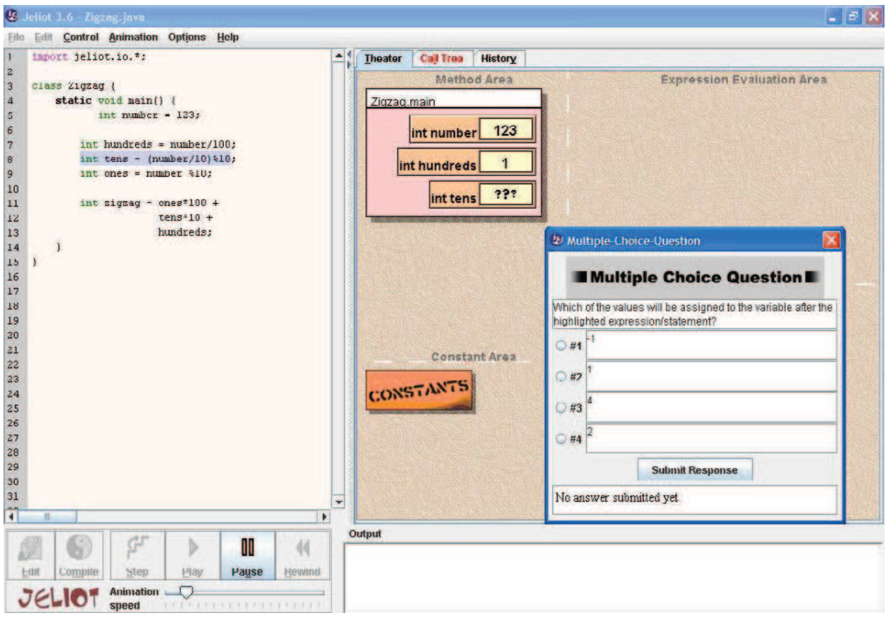
Fig. 1. The functional structure of Jeliot 3 [9].

- All the different concepts related to the expression that the question concerns. In this way, students performance in questions related to different concepts can be recorded into a user model.
- Correct answer that is the result of the program's interpretation.
- Zero to three incorrect answers depending on the question type: multiple choice questions have four answers, yes-or-no questions have two answers and open-ended questions have none. Currently, incorrect answers are determined randomly, but it could be possible to apply certain heuristics based on the previous steps in the execution and current values of variables.
- Location of the expression in the source code so we can highlight that part of the code when the question is shown.

As discussed in the previous section, the interpretation of the program produces a program trace that is called MCode. In order to collect the listed information, we implemented a *preprocessor for MCode*. It goes through the MCode before it is interpreted and extracts the information from the MCode. This information is saved so that MCode interpreter can query them based on the expression identifier during the interpretation. In the Figure 1, this phase would be located in the middle of the arrow 4.

Then during the interpretation of every MCode instruction, interpreter checks whether a question for the current instruction identifier is found. If a question is found, it is shown to the user before the instruction is animated. We adapted the *avInteraction* package developed by Rößling and Häussge [14] in order to present questions and collect users' answers. The answers can be saved into a file or a database. Thus, this feature can be used in order to adapt the program visualization as well as to summatively evaluate the students performance as part of their course work or on-line exam.

As a proof of concept, we have implemented a restricted question generation that only asks questions related to the results of the assignment statements. An example of the generated question is displayed in Figure 2. The question is shown in the right together with the visualization and the related code segment is highlighted in the left.



```
import jeliot.io.*;

class Zigzag {
    static void main() {
        int number = 123;

        int hundreds = number/100;
        int tens = (number/10)%10;
        int ones = number %10;

        int zigzag = ones*100 +
                     tens*10 +
                     hundreds;
    }
}
```

Which of the values will be assigned to the variable after the highlighted expression/statement?

#1 -1
#2 1
#3 4
#4 2

No answer submitted yet

Fig. 2. An example question generated by Jeliot 3.

Moreover, different kinds of question related to the execution and animation of the program can be automatically generated. We list here some possibilities:

- Predict the result of any expression evaluation.
- Ask the user to click on the variable which is part of the expression, or into which the result of an expression evaluation is going to be assigned.
- In loops, it is possible to ask if the execution will continue for next round or not and in conditional statements, it is possible to ask if the execution will continue into the `then` or `else` part of the statement.
- User can be asked to click the line (or line number) that is being executed next, for instance, after a method call or in the beginning of a loop or an if-statement.
- In sorting, it would be possible to determine a swap operation and ask the user to click on those array cells that are going to be swapped.

It is not feasible to pop up questions in every possible place, because students can get annoyed or tired of the questions. Thus, there should be ways to determine when it is most appropriate and meaningful to generate a question. For example, Jeliot could allow the user to select the variables or expression types that should generate questions and thus focus the questions on the selected concepts or parts of the program. Similarly to related systems (e.g. Problets and WadeIn (see Section 4)), we could adapt the question generation, visualizations and explanations based on the performance data of the user. We have done preliminary work on this direction, and it is described in [7].

Moreover, there should be a possibility for a teacher to manually create questions for programs in order to allow the use of question generation for quizzes or in-class exams. This can be achieved with the package that we use to display and save the question information, because it provides a file format for manual question specification [14].

4 Previous Work

Kumar et al. [4] have developed a system, called *Problets*, that generates exercises related to programming concepts (e.g. loops, pointers etc.) from language independent templates, thus supporting multiple programming languages. These exercises present a program and ask the user to identify the lines that generate output and determine what is the output during the execution of the program. In exercises regarding pointers, user needs to identify the code lines that are either syntactically or semantically erroneous. These exercises are delivered in the form of an applet that is connected to a server that handles the exercise generation and stores information related to the performance of the user. This is done in order to analyze what kind of exercises to present to the user.

When compared to the question generation in Jeliot 3, we can identify certain similarities and differences. Both ask questions related to the execution of a program. However, Problets are related to the program code, whereas questions in Jeliot can be related to the program code and visualization. This can give more variation in the question types as seen in Section 3. Jeliot supports dynamic aspects of the program execution, for example, user can give input to the program and the questions are adjusted accordingly because they are based on the information acquired during the interpretation process. Currently, Jeliot supports only Java. However, if interpreters for other programming languages are integrated into it, the question generation is language independent. Problets support multiple programming languages because of the language independent templates that are translated to the programming language in question. Problets can be used for learning and testing similarly as the automatic question generation in Jeliot 3.

Another related system is JHAVÉ [11] which combines visualization tools and support for interaction. It provides textual materials, questions, and other exercises related to the visualization, and thus engages users to the visualization. JHAVÉ supports only post-mortem visualization of the programs meaning that user needs to provide input data before the program or algorithm is run. The questions need to be defined manually into the source code of the program. These issues make the approach different from Jeliot.

WadeIn II [2] visualizes the expression evaluation in C language. The system consist of two modes: exploration and knowledge evaluation. The question generation is related to the knowledge evaluation mode in which student needs to demonstrate the understanding of the expression evaluation by simulating it. The task is to simulate the evaluation of the expression, whereas in Jeliot, a user is asked to predict what will happen next in the given context of the program and its execution.

5 Conclusion and Future Work

We have presented a way to automatically generate prediction questions during a program visualization automatically and a proof of concept implementation of it. We have also presented different types of questions that could be automatically generated with the same framework and ways to determine when those questions should be raised in order to support different ways of learning and testing.

As future work, we implement the proposed question types and test their usability. We also plan to study the use of question answering both during individual as well as collaborative learning of programming concepts and programming. We will variate the level of engagement to analyze its effects to the learning and collaboration. Furthermore, we can test how different types of questions support the understanding of programs and programming learning. For example, should the questions be related to data flow or control flow, or both. In addition to this, we are planning to use the feature in distance education as a part of the summative evaluation.

References

[1] Ben-Bassat Levy, R., M. Ben-Ari and P. A. Uronen, *The Jeliot 2000 program animation system*, Computers & Education **40** (2003), pp. 1–15.

[2] Brusilovsky, P. and T. D. Loboda, *WADEIn II: A Case for Adaptive Explanatory Visualization*, in: *Proceedings of The Eleventh Annual Conference on Innovation and Technology in Computer Science Education*, Bologna, Italy, 2006.

[3] Byrne, M. D., R. Catrambone and J. T. Stasko, *Evaluating animations as student aids in learning computer algorithms*, Computers & Education **33** (1999), pp. 253–278.

[4] Dancik, G. and A. Kumar, *A Tutor for Counter-Controlled Loop Concepts and Its Evaluation*, in: *Proceedings of Frontiers in Education Conference (FIE 2003)*, Boulder, CO, USA, 2003, pp. T3C–7–12.

[5] Evans, C. and N. J. Gibbons, *The interactivity effect in multimedia learning* (2006), accepted to Computers & Education.

[6] Hundhausen, C. D., S. A. Douglas and J. T. Stasko, *A meta-study of algorithm visualization effectiveness*, Journal of Visual Languages and Computing **13** (2002), pp. 259–290.

[7] Lin, T., A. Moreno, N. Myller, Kinshuk and E. Sutinen, *Inductive Reasoning and Programming Visualization, an Experiment Proposal*, in: *Proceedings of the Fourth International Program Visualization Workshop*, Florence, Italy, 2006, pp. 83–88.

[8] Moreno, A., "The Design and Implementation of Intermediate Codes for Software Visualization," Master's thesis, Deparment of Computer Science, University of Joensuu (2005), (http://cs.joensuu.fi/jeliot/files/Andres_thesis.pdf) (Accessed (5.9.2006)).

[9] Moreno, A., N. Myller, E. Sutinen and M. Ben-Ari, *Visualizing Program with Jeliot 3*, in: *Proceedings of the International Working Conference on Advanced Visual Interfaces, AVI 2004*, Gallipoli (Lecce), Italy, 2004, pp. 373–380.

[10] Myller, N., "The Fundamental Design Issues of Jeliot 3," Master's thesis, Deparment of Computer Science, University of Joensuu (2004), (`http://cs.joensuu.fi/jeliot/files/niko_thesis.pdf`) (Accessed (5.9.2006)).

[11] Naps, T. L., *JHAVÉ – Addressing the Need to Support Algorithm Visualization with Tools for Active Engagement*, IEEE Computer Graphics and Applications **25** (2005), pp. 49–55.

[12] Naps, T. L., J. R. Eagan and L. L. Norton, *JHAVÉan environment to actively engage students in Web-based algorithm visualizations*, in: *Proceedings of the thirty-first SIGCSE technical symposium on Computer Science Education* (2000), pp. 109–113.

[13] Naps, T. L., G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger and J. Á. Velázquez-Iturbide, *Exploring the Role of Visualization and Engagement in Computer Science Education*, in: *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education* (2002), pp. 131–152.

[14] Rößling, G. and G. Häussge, *Towards Tool-Independent Interaction Support*, in: A. Korhonen, editor, *Proceedings of the Third International Program Visualization Workshop*, Warwick, England, 2004, pp. 110–117.

P6.

6

Myller, N. and Nuutinen, J. (2006). JeCo: Combining program visualization and story weaving. *Informatics in Education*, 5(2):195–206.

Informatics in Education, 2006, Vol. 5, No. 2, 255–264
© 2006 *Institute of Mathematics and Informatics, Vilnius*

JeCo: Combining Program Visualization and Story Weaving

Niko MYLLER Jussi NUUTINEN
Department of Computer Science, University of Joensuu
P.O. Box 111, FI-80101 Joensuu, Finland
e-mail: niko.myller@cs.joensuu.fi, jussi.nuutinen@cs.joensuu.fi

Received: May 2006

Abstract. We present a collaborative learning tool for programming, Jeliot Collaboratively or JeCo. Jeliot Collaboratively is a combination of a program visualization tool for Java programs, called Jeliot 3, and a collaborative authoring tool, Woven Stories. We introduce these systems and explain how they can be used in learning. Furthermore, we present future directions in order to support a wider range of use cases with JeCo.

Key words: CSCL, program visualization, programming learning, collaborative authoring, Java.

1. Introduction

Currently, a large number of students are struggling with their programming courses (McCracken *et al.*, 2001) and especially in the distance education courses in programming (V. Meisalo *et al.*, 2003). Pair programming is one of the processes suggested by extreme programming (XP) methodology. Pair programming has been found successful in learning situations (Kuppuswami and Vivekanandan; McDowell *et al.*, 2003; Williams *et al.*, 2003; Nagappan *et al.*, 2003; Stotts *et al.*, 2003). Studies have shown that students enjoy programming in pairs, produce better quality programs and perform better in examinations. Furthermore, students should also learn team work skills because team work is one of the key elements of software development. Pair or group programming could be introduced to distance education programming courses as well but that would require new tools to support the interaction between students when they are learning and working at a distance. However, there have been only few attempts to develop appropriate tools (Ratcliffe and Thomas, 2004; Jehng and Chan, 1998; Hanks, 2005).

Hundhausen (Hundhausen, 2002) and Hübscher-Younger (Hübscher-Younger and Narayanan, 2003) have claimed that when communication and collaboration are accompanied by visualizations, students' learning of algorithms is enhanced. According to Hundhausen (Hundhausen, 2002), bi-directional communication between instructor and students as well as between students supported with relevant visualizations helps students to learn algorithmics because visualization supports relevant communication of the topic. Hübscher-Younger (Hübscher-Younger and Narayanan, 2003) states that construction, sharing and discussion of the students' representations of a certain algorithm help students to learn the algorithm better.

We have developed a program visualization tool, *Jeliot 3*, that can be used during learning and teaching of programming (Moreno *et al.*, 2004c). It is designed to help novice level students by visualizing the Java program's state and source code during the execution of the program. In this way, students are able to build a reference model that they can use to solve problems they have not encountered before and they acquire vocabulary to discuss programs with each other and their teacher (Ben-Bassat Levy *et al.*, 2003).

As a first attempt to support distributed collaborative programming and learning to program, we have combined a program visualization system, Jeliot 3, and a co-authoring tool, Woven Stories, into an application called *JeCo* (*Jeliot Collaboratively*) (Moreno *et al.*, 2004b; Moreno *et al.*, 2004a). The version of JeCo described in earlier papers was based on the previous version of Woven Stories which had limited capabilities. In this paper, we explain a new version of JeCo and the underlying systems: Jeliot 3 and the new version of *Woven Stories* (Nuutinen, 2006). Moreover, we present some future directions which could benefit all these systems in the contexts of distance education.

2. Previous Work

As stated in the previous section, new tools are needed in order to support the interaction between students when they are learning and working at a distance. There are several tools that support just the minimum requirements (e.g., collaborative editor or desktop sharing) to support distributed pair/group-programming as reviewed by (Hanks, 2005). However, there are less tools that try to elaborate on this and add other features to increase the usability and productivity of the tools such as visualizations or gestures (Hanks, 2005).

An example of a tool that uses visualization to support collaboration is *VorteX* (Ratcliffe and Thomas, 2004). It supports the design of programs with a modifiable class diagram that is shared by the group. This tool can be primarily used only during the design phase of programming, however, other parts of the programming process should also be supported. Jehng and Chan (Jehng and Chan, 1998) have developed a collaborative programming environment for LISP-LOGO. They also showed that their environment had a positive influence on learning results when students were learning recursion in collaboration. This is a good indication how visualization can support programming when students are not co-present. However, LISP-LOGO or LOGO based languages in general are not commonly used in programming courses and thus this approach is not widely applicable as such.

3. Jeliot

Jeliot 3 (Moreno *et al.*, 2004c) is a program visualization tool supporting teaching and learning of programming in introductory Java programming courses. It supports different kinds of approaches in teaching of introductory programming such as objects-first or fundamentals-first. It is released under the GPL and is freely available at (`http://cs.joensuu.fi/jeliot/`).

Fig. 1 illustrates the user interface of Jeliot 3. Most of the Java language concepts are currently supported and visualized by Jeliot 3. It displays the operation of a virtual machine during program execution. However, the animation takes place on the Java language level and not the level of bytecode to make it relevant for students. It shows all the details of the program execution by visualizing expression evaluations, method calls, and object- and array allocations. Thus, the visualization can be used to teach and learn programming concepts. Another view (not shown in the figure) shows the method call tree of the executed program. Furthermore, Jeliot provides a possibility to explore the history of the execution through static snapshots taken before and after a possible interesting event in the visualization.

In a classroom study, Ben-Bassat Levy *et al.* (Ben-Bassat Levy *et al.*, 2003) found that Jeliot 2000 supported the learning of mediocre students. Students created viable mental models of the program execution based on the Jeliot's visual display. Jeliot also provided them with vocabulary to describe the execution. Because of that students were able to answer questions related to unseen situations by drawing a Jeliot-like display and describing the situation through the diagram. Because Jeliot 3 uses the same visualization scheme as Jeliot 2000 and only extends it, we can assume that similar results are achieved when Jeliot 3 is used in a classroom.

One reason to develop Jeliot 3 was to enhance the modularity and extensibility of the previous versions. We have developed an extension that allows Jeliot 3 to interact with BlueJ, a widely used educational programming environment. It is possible to start animations in Jeliot directly from BlueJ's object bench. Another plugin has been developed for Editing Java Easily (EJE), a development environment containing special features that simplify the usage of the tool for novices and allows the use of the tool directly from a web page. Furthermore, in this paper we present how Jeliot 3 has been combined with a co-authoring tool, Woven Stories, to enable use of Jeliot collaboratively on the web.

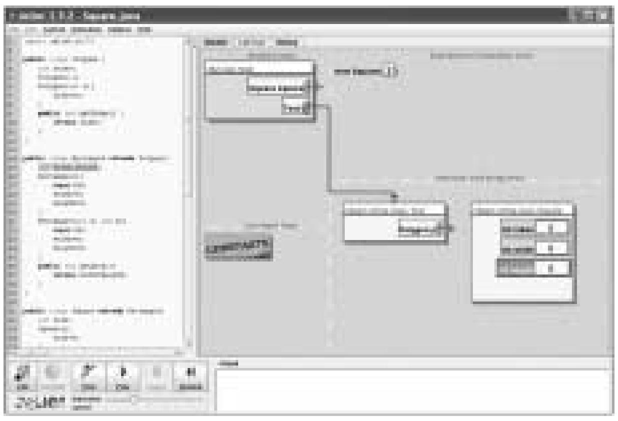


Fig. 1. User interface of Jeliot 3.

4. Woven Stories

Woven Stories (WS) (Gerdt *et al.*, 2001; Nuutinen, 2006) is a concept that allows persons to write stories with a new approach. The stories are built of small blocks that are called *sections*. Each section can contain an unlimited amount of text. The sections are linked together with arrows that are called *edges*. Using this approach the story can be visualized as a directed graph where each path forms an individual story.

Fig. 2 represents a simple woven story that has four different *storypaths*. These storypaths have been achieved by introducing three new sections to the original story of six sections (gray in Fig. 2). With this approach, the users are able to change the existing story by introducing new sections to it. It means that the users are able to introduce their ideas without removing anything from the original story. The result of the writing process with Woven Stories can be a versatile document with several optional storypaths.

The concept of Woven Stories is a mixture of *concept mapping* (Novak and Gowin, 1984), *flow charts*, *collaborative writing*, *graphs* and *finite automata*. It forces the users to think what they write, to divide the texts into sections and finally, to see the relationships between the sections. Since people learn what they process (Bereiter, 2002; pp. 274), this approach can be efficient for learners, especially in collaborative situations.

Based on the concept of the Woven Stories a client-server application has been developed (Nuutinen, 2006). This tool consist of a client, *the Loom*, and a server called *WS-Server*. Loom implements the concept of the Woven Stories into a collaborative environment, where synchronous and asynchronous users can work with the same story. Fig. 3 represents the user interface of the Loom.

While using the Loom the users can see at all times the structure of the document from the *Story Space* (1. in Fig. 3). The contents of the sections can be seen by selecting the section from the Story Space. The contents are displayed on the *Content Viewer* (2. in Fig. 3) at the top right corner. The Loom also supports small scale communication with *Chat* (4. in Fig. 3) and awareness with *Action Info* (5. in Fig. 3) that displays recent activities of the current document.

The Story Space has a relaxed WYSIWIS (What You See Is What I See) approach which means that the users have the same data, but do not necessarily see the same portion of it at all times. All the actions with the objects (sections and edges) are first sent to the server and then back to the clients. This guarantees that all the clients really have the same data at all times.

The Loom has been tested in a couple of situations with a small number of users, during which it was found that the concept is easy to understand and that the software is easy to use (Nuutinen, 2006). However, users found it problematic to reuse the sections that other people had written. Due to this the stories tend to have a sequential nature.

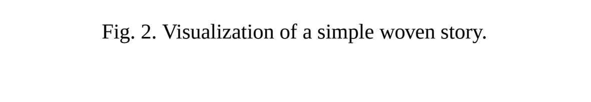


Fig. 2. Visualization of a simple woven story.

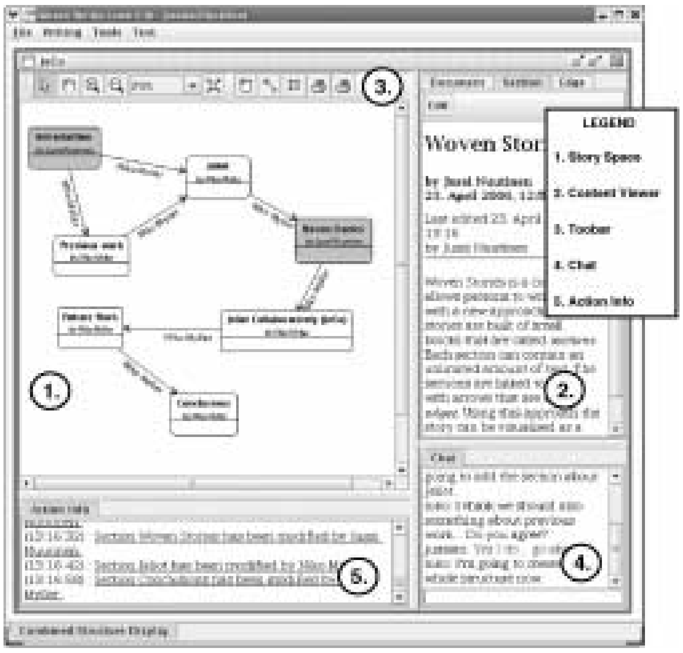


Fig. 3. User Interface of the Loom.

We believe that this approach can be efficient in various subject domains. If the communication and the awareness features of the Loom are improved, it can be used even more efficiently within distributed group.

5. Jeliot Collaboratively (JeCo)

Program visualization tools normally support only individual learning and engage students cognitively but do not necessarily support social activities, even though these social practices can be crucial to learning. In a classroom, the social activities can be handled with group activities that are guided by the teacher without any additional support from the software. However, in distance education this is not possible because the interaction and social aspects need to be supported by software. We want to support both cognitive and social processes in a distance education through *JeCo* (Moreno *et al.*, 2004b; Moreno *et al.*, 2004a), which is a combination of Jeliot 3 and Woven Stories.

JeCo gives students a possibility to collaborate with each other with tools that are made for the purpose. Students can acquire vocabulary related to the programming concepts from Jeliot and thus it creates a context for the discussions (Ben-Bassat Levy *et al.*, 2003). In the case of JeCo, this vocabulary will grow into an inter-subjective set of shared concepts.

Jeliot and Woven Stories are combined by modifying the Loom client program whereas the WS-Server can be kept as it is because of its flexible design. Fig. 4 illustrates

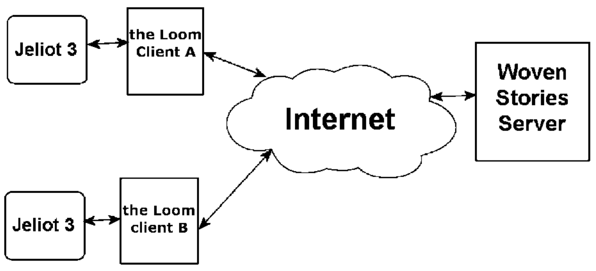


Fig. 4. The combination of Woven Stories and Jeliot in JeCo with multiple users.

how these systems are related to each other. The Loom client was modified to recognize whether a section contains program code that can be animated in Jeliot. If it does, the user is informed of this possibility by showing a text telling that Jeliot is available on that section. When a user clicks the right mouse button on that kind of section, there is an option in the popup menu to animate the program with Jeliot. This creates a link between these two systems and the animation allows both asynchronous and synchronous communication among students about a program or its visualization. The program code can be sent to the forum by editing a section and clicking on a button with Jeliot logo on it and paste the code into the appearing window. This will make the code attached to the section and recognizable to the Loom client.

For example, any user (e.g., *client A*) can send to the server a section that consists of a message and program source code. If another user (e.g., *client B*) wants to see the animation of the program code sent by *client A* and discuss that with *client A* about it, s/he can request the section and animate it. While Jeliot is animating the program, *clients A* and *B* can discuss the phases of the program's execution. This kind of scenario is illustrated also in Fig. 4 and the users would see a similar view as shown in Fig. 5. In the Fig. 5, the currently viewed section is bordered with a rectangle and sections containing the source code that can be animated with Jeliot 3 are indicated with the *Jeliot available* text in the bottom of the section rectangle. If the discussion leads into new developments, *Client B* could add another section to the document and connect it to the section(s) that inspi red this work. Thus, it would keep the history of the development visible to users that are not currently logged in into the system.

Because Woven Stories supports HTML documents, it is possible to bring the whole course materials inside multiple Woven Stories documents. For example, each topic of the course can be a separate document. These documents can contain regular text but also source code that can be visualized with Jeliot. Several students can then see the same visualization at the same time and discuss it, but it is also possible that students comment on the programs and give their own examples and let the other students try them out as well. Thus, the tool is supporting both synchronous and asynchronous modes of communication and learning. Moreover, JeCo can support programming courses as a platform in collaborative exercises, assignments and projects.

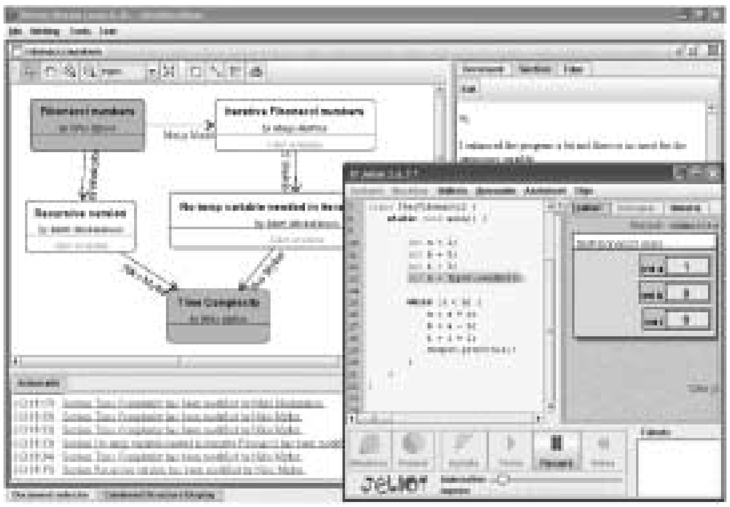

Fibonacci numbers
Iterative Fibonacci numbers
Recursive version
No temp variable needed in iterative
Time Complexity
JeLIOT

Fig. 5. User Interface of JeCo.

6. Future Work

As stated in Section 4, the users of the Loom have had problems to create stories with optional storypaths. While it is an important research question to find solutions to this problem within the development of Loom, it is not such an issue when considering JeCo. This is due to the fact that program versions tend to branch naturally. Furthermore, the storypaths can be used for other purposes than version management. For example, we could consider the story as a program and view the graph of sections as a class diagram of a program in which each section would be a class.

Currently, there are also some features missing from the Loom in order to support distributed use of the tool: Users should be more aware of other users and the communication capabilities should be enhanced.

Awareness (Gutwin and Greenberg, 1996) means the knowledge of other people and their actions. This is an important aspect of collaboration, since the users of the system must know what the others are doing at the same time, since this can affect their work. For example the users have to be aware who are present at the system in order to engage into conversations.

Communication is important part of collaboration. In systems where the users collaborate on-line, the communication should be carefully planned. Currently, the Loom supports communication only by chatting. While this can be used when the users are on-line, there is no way to communicate in other situations. Due to this, features supporting the asynchronous communication has to be created. For example features such as annotations (Nokelainen *et al.*, 2003) could be useful in context of Woven Stories. In synchronous communication, the users should be able to refer to the artifacts they are dealing with. For example, in a chat session a user might want to refer to a just written section and want to get opinions from others. In order to make things simple, users should be able to point at that section from the Story Space. Such features are called *conversational props* (Brinck and Gomez, 1992).

By implementing the features to improve the support of awareness and communication in the Loom, we can improve the usefulness of the Loom in distance use cases. When paying attention also to asynchronous collaboration aspects, the Loom can be used also in situations where the users are not able to be on-line at the same time. Since the JeCo is based on the Loom these same benefits apply also to it. For example, the annotation could be used to comment the source code or similar code segments in multiple files could be annotated in order to find similarities.

Currently, the visualizations of two distributed users are not synchronized, and thus it can be difficult to discuss a very specific point in a visualization. There should be a possibility to stream one visualization to multiple users so that one user would be the host of the visualization and other users could see the visualization but not necessarily control it. This could be used also for guided session where a teacher controls the visualization and send explanations to the chat and students can see the visualization from their screens and post questions and comments to the chat.

7. Conclusion

In this paper, we have introduced a program visualization tool, Jeliot 3, a collaborative authoring tool, Woven Stories, and their combination JeCo, a collaborative learning tool for programming. The combination was possible due to their flexible and modular design. We also discussed the possible scenarios where these systems can be used and how. We have also presented future directions in order to enhance the learning experience with JeCo. We have already published Jeliot 3 under GPL license and plan to do so for JeCo as well. We hope this will encourage others to use the tools and contribute to them in order to form a learning community of users and developers.

References

Ben-Bassat Levy, R., M. Ben-Ari, P.A. Uronen (2003). The Jeliot 2000 program animation system. *Computers & Education*, **40**(1), 15–21.

Bereiter, C. (2002). *Education and Mind in the Knowledge Age*. Lawrence Erlbaum Associates, Mahwah, NJ.

Brinck, T., and L.M. Gomez (1992). A collaborative medium for the support of conversational props. In *CSCW '92: Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work*. ACM Press, New York, NY, USA. pp. 171–178.

Gerdt, P., P. Kommers, C.-K. Looi, E. Sutinen (2001). Woven stories as a cognitive tool. In *Cognitive Technology: Instruments of Mind, Lecture Notes in Computer Science*, **2117**, 233–247.

Gutwin, C., and S. Greenberg. Workspace awareness for groupware. In *CHI '96: Conference Companion on Human Factors in Computing Systems*. pp. 208–209.

Hanks, B. (2005). *Tool Support for Distributed Pair Programming – Empirical Studies of Distributed Pair Programming*. PhD dissertation, Computer Science, University of California, Santa Cruz. `http://faculty.fortlewis.edu/hanks_b/publications/dissertation.pdf` (Accessed 8.8.2006).

Hübscher-Younger, T., and N.H. Narayanan (2003). Constructive and collaborative learning of algorithms. *SIGCSE Bulletin*, **35**(1), 6–10.

Hundhausen, C.D. (2002). Integrating algorithm visualization technology into an undergraduate algorithms course: ethnographic studies of a social constructivist approach. *Computers & Education*, **39**(3), 237–260.

Jehng, J.-C.J., and T.-W. Chan (1998). Designing computer support for collaborative visual learning in the domain of computer programming. *Computers in Human Behavior*, **14**(3), 429–448.

Kuppuswami S., and K. Vivekanandan (2004). The effects of pair programming on learning efficiency in short programming assignments. *Informatics in Education*, **3**(2), 251–266.

McCracken, M., V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. Ben-David Kolikant, C. Laxer, L. Thomas, I. Utting, T. Wilusz (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students, *SIGCSE Bulletin*, **33**(4), 125–180.

McDowell, C., L. Werner, H.E. Bullock, J. Fernald (2003). The impact of pair programming on student performance, perception and persistence. In *Proceedings of the 25th International Conference on Software Engineering*. IEEE Computer Society. pp. 602–607.

Meisalo, V., E. Sutinen, S. Torvinen (2003). Choosing appropriate methods for evaluating and improving the learning process in distance programming courses. In *Proceedings of the 33rd ASEE/IEEE Frontiers in Education Conference (FIE2003)*. Boulder, CO, USA. pp. T2B-11–16.

Moreno, A., N. Myller, E. Sutinen (2004a). Collaborative program visualization with woven stories and Jeliot 3. In *Proceedings of the IADIS International Conference on Web Based Communities 2004*. Lisbon, Portugal. pp. 482–485.

Moreno, A., N. Myller, E. Sutinen (2004b). JeCo: a collaborative learning tool for programming. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*. Rome, Italy. pp. 26–29.

Moreno, A., N. Myller, E. Sutinen, M. Ben-Ari (2004c). Visualizing programs with Jeliot 3. In *Proceedings of the International Working Conference on Advanced Visual Interfaces AVI 2004*. Gallipoli (Lecce), Italy. pp. 373–376.

Moreno, A., N. Myller, E. Sutinen (2004b). JeCo: a collaborative learning tool for programming. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*. Rome, Italy. pp. 26–29.

Nagappan, N., L.A. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, S. Balik (2003). Improving the CS1 experience with pair programming. *SIGCSE*, 359–362.

Nokelainen, P., J. Kurhila, M. Miettinen, P. Floren, H. Tirri (2003). Evaluating the role of a shared document-based annotation tool in learner-centered collaborative learning. In *Advanced Learning Technologies, Proceedings of the 3rd IEEE International Conference*. pp. 200–203.

Novak, J.D., and B.B. Gowin (1984). *Learning How to Learn*. Cambridge University Press, New York.

Nuutinen, J. (2006). Designing a computer supported collaborative mindtool: woven stories. *Licentiate Thesis* (Manuscript in preparation).

Ratcliffe, M.B., and L.A. Thomas (2004). Understanding our students: incorporating the results of several experiments into a student learning environment. In *16th Workshop of the Psychology of Programming Interest Group*. Carlow, Ireland. pp. 10–20.

Stotts, P.D., L.A. Williams, N. Nagappan, P. Baheti, D.Jen, A. Jackson (2003). Virtual teaming: experiments and experiences with distributed pair programming. In F. Maurer and D. Wells (Eds.) *XP/Agile Universe, Lecture Notes in Computer Science*, **2753**, pp. 129–141.

Williams, L.A., C. McDowell, N. Nagappan, J. Fernald, L.L. Werner (2003). Building pair programming knowledge through a family of experiments. *ISESE*, 143–153.

N. Myller has a master’s degree in mathematics and he has been working with educational technology since 2001. He has worked in number of interdisciplinary international research projects. He is currently employed as a senior assistant at the Deparment of Computer Science at the University of Joensuu and studying for his PhD under supervision of professor Erkki Sutinen (University of Joensuu, Finland) and Dr. Piet Kommers (University of Twente, the Netherlands). His research topic is collaborative editors and mindtools and especially a software called Woven Stories.

J. Nuutinen received his BSc in 2003 and MSc in 2004 both from the Department of Computer Science at University of Joensuu in a record time of 2.5 years from starting. Currently, he is studying for his PhD under supervision of prof. Erkki Sutinen (University of Joensuu) and prof. Mordechai Ben-Ari (Weizmann Institute, Israel) and the expected graduation is in 2007. His research interests lie in the fields of visualization and concretization technologies, CSCL, information retrieval, computer ethics and adaptive systems. He has published more than 30 papers in international journals and conferences.

Priemonė „JeCo“: programų vizualizavimo ir fabulos dėstymo jungtis

Niko MYLLER, Jussi NUUTINEN

Straipsnyje supažindinama su programavimo mokymo priemone „Jeliot Collaboratively“ arba „JeCo“. „Jeliot Collaboratively“ apima programų vizualizavimo priemonę „Java“ programoms „Jeliot 3“ ir interaktyvią priemonę „Woven Stories“, skirtą kuriantiems rašinius autoriams. Straipsnyje aprašomos šių sistemų galimybės ir supažindinama su jų panaudojimo mokymesi būdais. Be to, brėžiamos kryptys ateičiai – siekiama išplėsti „JeCo“ galimo panaudojimo galimybes.

Dissertations at the Department of Computer Science and Statistics

Julkaisija	Joensuun yliopisto Tietojenkäsittelytieteen ja tilastotieteen laitos
Publisher	University of Joensuu Department of Computer Science and Statistics
Sarjan toimittaja Series Editor	Erkki Sutinen
Vaihdot	Joensuun yliopiston kirjasto/Vaihdot PL 107, 80101 Joensuu Puh. 013-251 2677, fax 013-251 2691 email: vaihdot@joensuu.fi
Exchanges	Joensuu University Library/Exchanges P.O. Box 107, FI-80101 Joensuu, FINLAND Tel. +358-13-251 2677, fax +358-13-251 2691 email: vaihdot@joensuu.fi
Myynti	Joensuun yliopiston kirjasto/Julkaisujen myynti PL 107, 80101 Joensuu Puh. 013-251 4509, fax 013-251 2691 email: joepub@joensuu.fi
Sales	Joensuu University Library/Sales of publications P.O. Box 107, FI-80101 Joensuu, FINLAND Tel. +358-13-251 4509, fax +358-13-251 2691 email: joepub@joensuu.fi